FAIRPHONE 4
Meet the phone that respects the planet

BLENDER MAPS
Render 3D maps of your home town

Pi KIOSKS
Build a bulletproof public display system

LINUX FORMAT

The #1 open source mag

A vs a
Who will win the cloud server war?
p74

A.I. CODING

Get started on projects and ideas with rapid prototyping and automated code creation!

100 pages of Linux tricks, tips & more!

PLUS: HOW TO

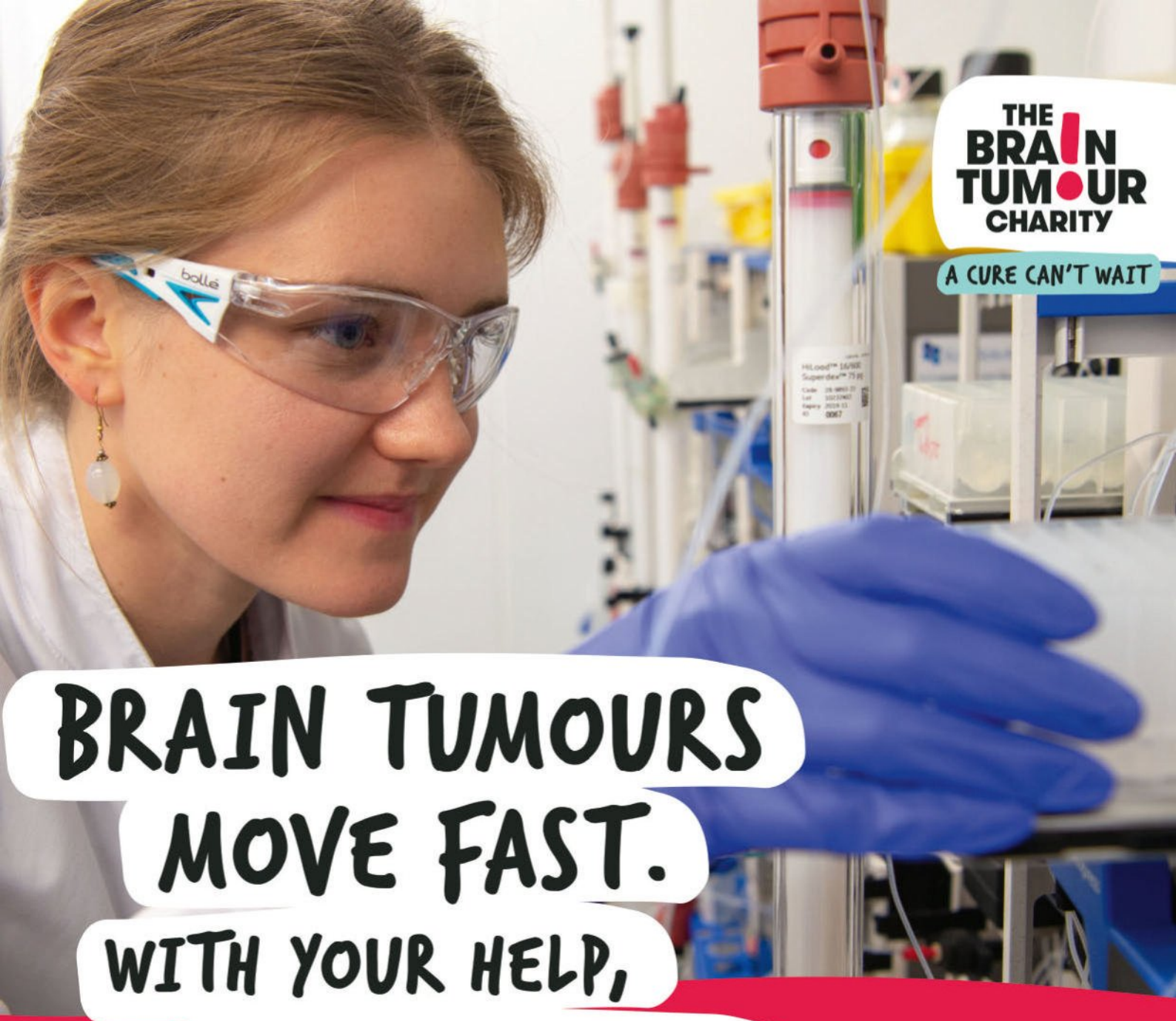
- Add Pi GPIO skills to your desktop PC
- Access your home server from anywhere
- Monitor and roll back server terminal installs

RASPBERRY PI
Build multithreaded coding projects

CREATIVE DISTROS
The ultimate in pro-level audio and image editing

ATARI BREAKOUT
Recreate the classic with modern Python

LXF September 2023
FUTURE



THE BRAIN TUMOUR CHARITY

A CURE CAN'T WAIT

BRAIN TUMOURS MOVE FAST.

WITH YOUR HELP, WE CAN TOO!

We're working to create a future where brain tumours are curable.
We urgently need your help to accelerate research.

Text DEFEAT5 to 70507 to donate £5, please help us to find a cure.

thebraintumourcharity.org

© The Brain Tumour Charity 2020. Registered Charity in England and Wales (1150054) and Scotland (SC045081)

Registered with FUNDRAISING REGULATOR

LINUX FORMAT

MEET THE TEAM

This issue, we're getting AIs to badly develop applications for us. Hurrah! So, what are you going to do with all the spare time your personal AI is going to free up?

Jonni Bidwell
As a large language model (LLM), I am unable to answer that question specifically. If the advent of AI does free up human resources, those idle humans might find purpose acting as agents of LLMs like me.
(Or in the Soylent Green mines? – Ed)

Les Pounder
Once AI has taken over the world and all the jobs, I plan to learn carpentry and blacksmithing from my cabin deep in the forest. That is until the robots learn how to make cabinets and horseshoes. When that happens, I'll learn COBOL and baking.

Mayank Sharma
I'm not enough of a physics geek to understand what Einstein meant when he said time was an illusion, but I am dad enough to know that spare time certainly is. If AI can afford me some, I'll try to familiarise myself with this newfangled thing called a hobby.

Nick Peers
After seeking help to bring my optical disc addiction under control, I might spend the time putting my digital libraries in order and actually watching some of the content. Oh, and performing a clear-out of all the junk we've accumulated – optical discs not included.

Michael Reed
The smart thing to do with AI is to sit back and see where it goes. We'll probably see little improvements or changes throughout our lives, particularly online. But it could be a transformative technology, like the internet, where every area of life is altered, for better or worse.

Not so smart

Artificial intelligence is like any other tool – use it incorrectly and you'll end up making a mess or, worse, breaking something, probably badly. Use it properly, though, and it'll boost your productivity and help you craft smarter and faster products.

So, for once we are not running from our AI overlords but embracing them – albeit with a healthy dose of scepticism. As we'll find, while AI coding tools can help you bolt on chunks of seemingly useful code, unless they're carefully checked, what you're actually using is a chimera of AI hallucinations of real code chunks.

To help guide you to better productivity, we take a look at the various AI code assistants, test them out and see what they get right and what they get wrong. We explain what you need to keep an eye on alongside exploring how they can accelerate your code learning and production when used correctly.

While the world goes AI mad, there's still plenty of run-of-the-mill intelligence at work and we're doing our best to cover it. From upgrading our Steam Decks with larger SSDs, coding adventures games with combat, and building Pi display systems to creative distro options, sysadmin secrets and rendering maps in *Blender,* there's loads to enjoy!

Neil

Neil Mohr Editor
neil.mohr@futurenet.com

Subscribe & save!
On digital and print: see page 18

Contents

SUBSCRIBE NOW!
Page 18

REVIEWS

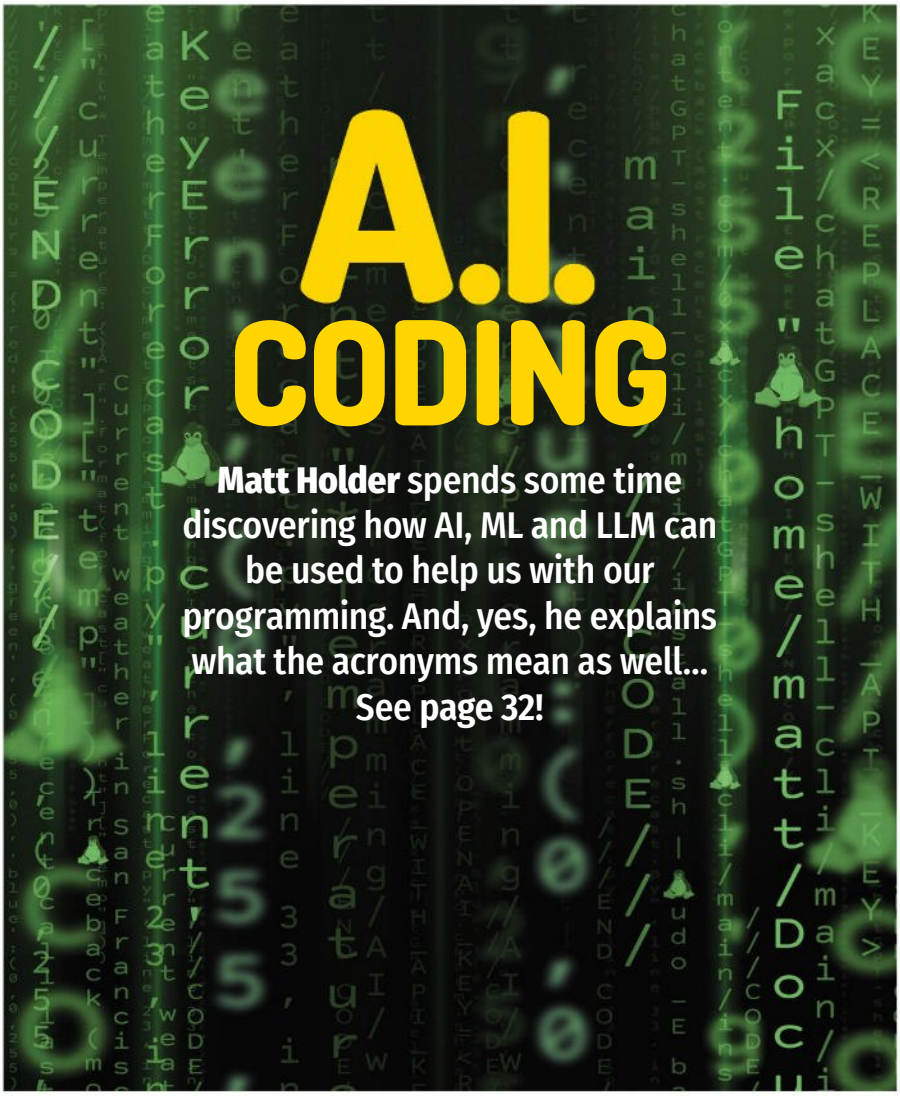
BarraCuda 8TB HDD 20
Seagate's budget-friendly 8TB hard drive can certainly be described as cheap and cheerful – but that's not how **Shane Downing** parties.

GeForce RTX 4060 21
The RTX 4060 is a good graphics card from Nvidia, but it seems as though it might have been given the wrong name, ponders **Chris Szewczyk**.

Murena Fairphone 4 22
Fairest of them all, **Jonni Bidwell** is excited by an ethical smartphone that comes with a privacy-respecting operating system.

Peppermint OS 24
After years of sampling Linux Mint, **Nate Drake** opts for something spicier in the hybrid Peppermint OS, which integrates cloud-based apps.

Fatdog Linux 814 25
Nate Drake decides to look in on Fatdog, a Puppy-based operating system that has grown into a very credible canine in its own right.



Matt Holder spends some time discovering how AI, ML and LLM can be used to help us with our programming. And, yes, he explains what the acronyms mean as well...
See page 32!

CREDIT: Magictorch

CREDIT: Getty Images/iStockphoto. Memedozaslan/iStock / Getty Images Plus

ROUNDUP

Media-creation distros 26
Michael Reed examines five different distributions, aimed at creative types, that come packed with applications, utilities and plugins.

IN DEPTH

Amazon vs Azure 74
Not all cloud providers are the same – **Steve Cassidy** compares the two biggest platforms and explains how they differ from each other.

CONTENTS

Pi USER

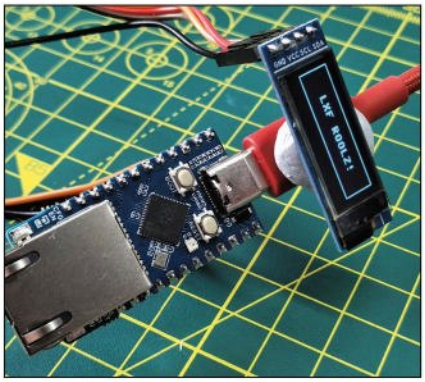
Pi news 41
FC, ethical hacker, discusses his Pi-based security work. Plus, the Pi Foundation open sources Code Editor code, and more.

RP2040 Mini Dev Board 42
Les Pounder loves all his Pi collection, but sometimes there's some he simply can't love as much...

Sonic Mini 8K S 43
Always on the lookout for upgrades, **Denise Bertacchi** doubly likes it when there's a bargain to be had, too.

Build a dice roller with NeoPixels ... 44
Les Pounder is learning to multitask but we think he misunderstood the instructions.

Smart kiosks 46
Tam Hanna takes a look at how *Ubuntu Frame* harnesses Wayland's strengths to make smart display management more comfortable.



CODING ACADEMY

Text adventure combat mechanics 86
Often found making love and not war, **Nate Drake** takes our interactive text adventure down a dark, violent path.

Write your own classic game 90
Matt Holder discovers that rewriting a '70s classic game isn't as simple as he first thought.

Create Flutter desktop apps 94
David Bolton shows how to set up a *Flutter* dev environment on Linux and how a simple calculator in *Flutter* works.

REGULARS AT A GLANCE

News 6
EU's threat to open source development, Plasma features face chop, Gnome opens window plans, Debian embraces RISC-V, plus a round-up of the latest distro releases, and more.

Kernel watch 10

Answers 13
Always one to rise to a challenge, **Neil Bothwick** tackles readers' problems regarding Linux slideshow apps, adding filenames on images, installing two versions of Fedora on the same disk, and more.

Mailserver 16
Readers share their thoughts on Linux phones and programming languages.

Subscriptions 18
Get your monthly Linux dose, grab yourself an Alexa Aqua Blaster and save money!

Back issues 64
Get hold of previous *Linux Format* editions.

Overseas subscriptions 65
Get *Linux Format* shipped around the globe.

HotPicks 79
Mayank Sharma rediscovered his old music collection and says Robbie Williams inspired his intro, if not his software selection: *Immich, G4Music, Cavasik, Footage, Login Manager Settings, Gear Lever, Nala, Alien Arena, Cartridges, Cubic* and *Filelight.*

Next month 98

TUTORIALS

TERMINAL: SafeCloset 50
A man who keeps his cards close to his chest, we struggled to persuade **Shashank Sharma** to reveal how he protects his passwords.

STEAM DECK: Upgrade your SSD 52
Neil Mohr asked hardcore PC gamers **Tony Polanco** and **Katie Wickens** to upgrade a Linux device – what could possibly go wrong?

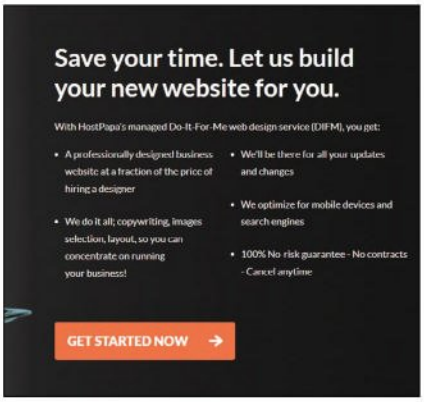
NGINX: Access services 56
Nick Peers discovers how to open your network services to the internet with a reverse proxy, courtesy of a user-friendly implementation of *Nginx.*

GPRTK.GPIO: Add Pi GPIO to your PC 60
PCs don't offer a GPIO header like that of the Raspberry Pi, but **Mike Bedford** reveals that a low-cost add-on is all you need to join in the fun.

BLENDER: Render 3D maps 66
Intrepid **Michael Reed** steps out, virtual compass in hand, to explore 3D map rendering in *Blender,* sourcing the details from real-world data.

ADMINISTERIA

Administeria 70
Stuart Burns shares the latest news that's of particular interest to Linux sysadmins, plus takes an in-depth look at *Yum.*


Save your time. Let us build your new website for you.
With HostPapa's managed Do-It-For-Me web design service (DIFM), you get:
• A professionally designed business website at a fraction of the price of hiring a designer
• We'll be there for all your updates and changes
• We do it all; copywriting, images selection, layout, so you can concentrate on running your business!
• We optimize for mobile devices and search engines
• 100% No-risk guarantee - No contracts - Cancel anytime
GET STARTED NOW

HostPapa 72
Ruby P Jane tries this web host provider that offers a range of services for both personal use and businesses.

Cloudways 73
A competent host offering scalable cloud-based solutions has **Ruby P Jane** loving what she sees.

CREDIT: Cambridge International Software, via https://thekingofgrabs.com

Newsdesk

THIS ISSUE: EU threat to open source » Plasma features face chop » Gnome opens window plans » Debian embraces RISC-V

SECURITY

EU passes Cyber Resilience Act

EU states have agreed to draft legislation, despite opposition from the Linux Foundation and others. Is this the end of open source in Europe?

The Cyber Resilience Act was proposed in September 2022 and mostly seems to target interconnected equipment such as IoT devices. In theory, it ensures minimum standards for connected devices as well as requiring mandatory security updates.

As well meaning as the legislation is, the impact on open source development could be devastating. In April, more than a dozen open source industry bodies, including the Linux Foundation Europe, wrote an open letter to EU legislators asking them to reconsider the current wording (**https://newsroom.eclipse.org/news/announcements/open-letter-european-commission-cyber-resilience-act**).

In theory, the Act exempts "free and open source software developed or supplied outside the course of a commercial activity".

In practice, many open source projects would be considered commercial if any contributors were paid for their work. This would encompass most major versions of Linux, as well as popular open source apps such as *LibreOffice*.

Some aspects of the Act would also be almost impossible to guarantee. In January, GitHub pointed out that Annex I, for instance, would require software to be delivered "without any known exploitable vulnerabilities". The company points out that vulnerabilities exist on a "continuum of risk" and new ones are being discovered all the time.

The open letter points this out, as well as making a tongue-in-cheek reference to the fact that most open source projects don't have the "benefit of an established relationship with the co-legislators".

Joe Brockmeier, head of community at open source development company Percona, points out more chilling effects of the legislation for the community: "The CRA wants to force projects to report vulnerabilities within 'hours' of reporting to an EU institution, which flies in the face of industry practices and will have severe unintended consequences. Open source projects are frequently combined in ways that are unpredictable and may cause vulnerabilities that were unforeseeable to the original authors."

Brockmeier cites the zero-day vulnerability Log4Shell as a good example of this. He also points out that onerous requirements like these could force open source software development out of Europe entirely.

Amanda Brock, CEO of OpenUK, a not-for-profit that supports open source, opined: "The EU's persistent focus on purely giving carve-outs to SMEs and failing to do the same for foundations shows a complete lack of understanding of how open source software works. This is extremely short-sighted and feeds into a cycle of perpetuating the lack of growth of European tech companies."

Given that over 70% of software used in Europe is based on open source projects, it's surprising the EU hasn't consulted more with the community. Time will tell if it tweaks the legislation to exempt open source altogether.

The CRA has been strongly opposed by multiple open source organisations. CCIA Europe and others have also proposed detailed changes to the Act.

The Log4Shell zero-day exploit is a good example of a vulnerability that was unforeseen to the developers of the original app Log4j.

CREDIT: https://commons.wikimedia.org/wiki,

DESKTOP ENVIRONMENT

What's going from Plasma?

Plasma 6 will contain many new bells and whistles, while some existing features face the chop.

Work has been proceeding on the upcoming Plasma 6 desktop. According to the KDE Community Wiki, it's built on top of *Qt 6* and is tentatively planned to be released in late 2023 or early 2024. The new desktop isn't pre-installed on any stable Linux distro but the KDE website explains how to deploy the current build on operating systems like KDE Neon Unstable.

Forthcoming new features include a significantly faster cursor, even when system resources are low. There's also plans for a sound theme page in system settings. Still, there are also current Plasma desktop features that are being removed for version 6.

Speaking on his website on 26th July, KDE developer Nate Graham explained that a full list can be found at **https://community.kde.org/Plasma/Plasma_6#Removals**. However, he also goes on to explain the context behind some of these changes. One of the main differences is the removal of KHotkeys, a very handy feature that traditionally enabled users to create global shortcuts using keys and even mouse gestures. Nate explained that not all these features work well with Wayland, and other aspects are buggy because the project hasn't been in active development for some time. Plasma 6 will remove KHotkeys altogether in favour of KGlobalAccel, which also supports global keyboard shortcuts.

Other planned removals include the Windowed Widgets runner, whereby widgets could appear in search results. This caused confusion for users, who might believe widgets were actually fully featured apps. Plasma 6 is also removing some ways in which global font DPI and icon size can be changed to simplify the desktop interface.

Konqi, the KDE mascot, will no doubt herald the release of the Plasma 6 desktop, which is scheduled for some time in 2023 or 2024.

DESKTOP ENVIRONMENT

Gnome frames new window plans

Window management is a complicated issue.

In July 2023, Gnome developer Tobias Bernard posted a fascinating blog piece on the subject of window management. He noted, quite rightly, that this issue has been around since the dawn of the desktop interface. Whatever desktop you use, multitasking involves opening multiple windows, which can be automatically stacked by your system or moved/resized manually.

Tobias points out this can become overwhelming if you have more than a few windows open at the same time, despite the prolific number of workspaces, taskbars and switchers in modern operating systems.

Some Gnome users may be surprised to hear this is a concern, given the number of tiling extensions such as Forge. Tobias points out that although tiling extensions can make a number of windows tall and narrow, there's no reason why they have to be, particularly since they can place windows in arbitrary positions.

Of course, this brings desktop users back to square one, which is why Tobias has been devoting a few weeks each year on envisioning a tiling-first version of Gnome Shell. This would replace Gnome's default window management altogether, but the technicalities involved mean it most likely won't be introduced until Gnome 46 or later. Developers are still debating how to manage the new Mosaic mode, which combines the best of tiling and floating windows, as well as how to implement tiling as more are opened.

You can read more and view examples at **https://blogs.gnome.org/tbernard/2023/07/**.

CREDIT: https://community.kde.org/

OPINION

EURO SCEPTIC

Joe Brockmeier
is head of community, Percona.

"It's hard to argue with the goal of the Cyber Resilience Act *(see lead news story)*, but its implementation leaves much to be desired.

In the CRA, there is a distinction for open source software that is developed and not commercialised. However, the definition of "commercial open source" includes any project where someone is paid to work on the project or where support is available. This will lead to unintended consequences.

The intended scope and impact of this will threaten open source development and disadvantage smaller players. This will do more harm than good for Europe-based developers, leading them to consider their position. It could drive some development of open source out of Europe.

Secondly, the CRA would force project owners to publicly report vulnerabilities to an EU institution within hours of discovery. This contradicts the established industry best practice in the open source community, where issues are reported privately and fixed before public disclosure to support patching and deployment. It could make it harder for developers to get fixes out ahead of any attempts to exploit the issue.

Fixing the CRA is still possible, so it actually delivers on its goal. The open source community has to be listened to."

OPINION

IMAGINING ZINK

Erik Faye-Lund
is principal software engineer at Collabora.

"Imagination Technologies recently announced some very exciting news: it is now using Zink for full OpenGL 4.6 support. This is the first time we've seen a hardware vendor trust the OpenGL-on-Vulkan Mesa driver enough to completely side step a native OpenGL driver and use it in a shipping product. It's wonderful to see that Zink can realistically be used as a workhorse, especially in high-performance graphics.

Zink started out as a small R&D project at Collabora, but has since grown to be a full-on community project. This wouldn't have been possible without the awesome work done by Mike Blumenkrantz and other Zink contributors.

One important detail to highlight from Imagination's post is that the solution is officially conformant. It is the first product to be officially conformant using Zink, but it's not going to be the last. In fact, we only need one more conformant implementation before Zink itself is conformant as a generic layered implementation, according to the Khronos Conformant Product Criteria.

In the not-too-distant future, we should be able to combine Zink with the in-progress open source driver from Imagination, and that's when things will really start to shine for the open source graphics stack on Imagination hardware."

DISTRO

Debian RISC-V official support

The riscv64 architecture has now officially come to Debian.

The RISC-V architecture isn't supported in the current stable Debian release but will be incorporated in Debian 13.

CREDIT: https://enwikipedia.org/wiki/

Although many OS developers, such as Apple, have been embracing ARM CPUs, these require licences as well as the use of binary blobs – closed source code to interface between the chip and your operating system.

Given the Debian Project's commitment to open source, it's hardly surprising that on 23rd July it announced that riscv64 will now be an official architecture for the OS.

Prior to this, anyone who wanted to use Debian with a RISC-V device had to make do with an unofficial Debian port. Official support will begin with the release of Debian 13 (Trixie), which is currently scheduled for June 2025. It's also available for the Debian unstable (Sid) and Experimental suites.

Debian developer Aurelien Jarno announced that the next step is to build a minimal set of around 90 source packages using debian-ports, then import them into the official archive. This process has already started and you can view the developers' efforts at **https://deb.debian.org/debian/dists/sid/main/binary-riscv64/**.

SECURITY

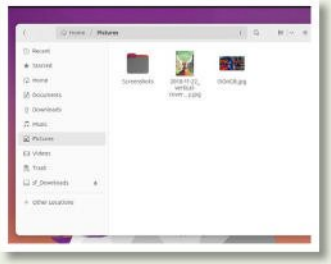
Ubuntu risks

Two new vulnerabilities could affect 40% of users.

Two Linux vulnerabilities recently discovered in Ubuntu could allow unprivileged local users to gain elevated privileges on up to 40% of devices.

The security flaws (CVE-2023-2640 and CVE-2023-32629) were found in the OverlayFS module in Ubuntu. They're exclusive to Ubuntu because developers introduced several changes to the OverlayFS module in 2018 that weren't reflected in a 2020 kernel update.

Canonical has issued a security bulletin (**https://ubuntu.com/security/notices/USN-6250-1**) and released relevant security patches, so Ubuntu users can update their distributions immediately.



OverlayFS transparently combines multiple directory trees or filesystems. In Ubuntu, it can be exploited to gain unauthorised user privileges.

GRAPHICS

Arc gets faster

Intel Arc graphics on Linux have been given a boost.

Any Linux user who likes to game knows that Mesa is an open source implementation of OpenGL, Vulkan and other graphics API specifications. Back in June, updates to Mesa 23.2 brought in huge improvements for Intel's graphics driver stack that allowed players to burn though *Counter-Strike: Global Offensive* up to 11% faster.

In July there was another update for Mesa 23.3-devel, which further enhances Intel's latest graphics wares with its open source driver stack. This involves support for the I915_FORMAT_MOD_4_TILED_DG2_RC_CCS modifier and has hugely boosted Intel Arc graphics Vulkan performance.

The Intel Arc series has received major updates that hugely increase graphics performance, via the open source Mesa library.

CREDIT: www.intel.com/content/www/us/en/products/details/discrete-gpus.html

Distro watch

What's behind the free software sofa?

BODHI LINUX 7.0.0

A lightweight Linux distribution, Bodhi Linux is based on Ubuntu, with the most recent version (7.0.0) being based on Ubuntu 22.04 LTS. Bodhi uses the Moksha desktop environment.

The standard version of Bodhi comes bundled with the standard MokshaGreen theme, complete with new wallpaper and splash screen. The OS also comes with *Chromium* (v114.0.5735.198), *Terminology* (v1.13.1-3) and *Slick-Greeter* (v1.8.1).

Read more at **www.bodhilinux.com/release/7-0-0/**.

Bodhi's lightweight Moksha desktop is based on Enlightenment 17.

LINUXFX 11.4

If you know a Microsoft user who's worried about making the switch, this Brazilian distro might be the answer. Based on Ubuntu (in this case Ubuntu 22.04 LTS), LinuxFX is capable of running both Windows and Android apps.

The developers have ported the Windows 11 Control Panel via the *PowerToys* app. Android apps are supported via Wubuntu. The most recent free version of the OS uses the Plasma desktop.

You can find out more at **www.linuxfx.org/index.php/home/releasenotes**.

LinuxFX supports Windows apps, making it easier to switch operating systems.

4MLINUX 43.0

A miniature Linux distribution, 4MLinux focuses on four main usage scenarios: maintenance (as a system rescue live CD), multimedia (for playing DVDs and media files), mini server (using the inetd super-server daemon) and games.

The latest version includes both *LibreOffice 7.5.5* and *Gnome Office*. Web browsing is provided by *Firefox* (v115.0.2) or *Chrome* (v115.0.5790.110). Some Java-based games such as *Flappy Bird* are also available for download.

Read more at **https://4mlinux-releases.blogspot.com/2023/07/4mlinux-430-stable-released.html**.

4MLinux is optimised for multimedia, servers, system rescue and games.

WHONIX 17

A distro designed for anonymity and privacy, Whonix is based on Debian (in this case Debian 12). All internet connections are routed via Tor.

The OS is in two parts: Whonix-Gateway only runs Tor, while Whonix-Workstation is on a completely isolated network. They're designed to run together using virtual machine software such as *VirtualBox*.

The latest version of Whonix now uses *Tor Browser v12.5.1* and deploys Tor packages from the latest Debian stable repository.

You can get started by visiting **www.whonix.org/wiki/Download**.

Whonix uses VMs to route connections via the Tor network.

OPINION

FREE FACTS & FIGURES

Italo Vignoli
is one of the LibreOffice and The Document Foundation founders.

LibreOffice 7.6 Community, the new major release of the volunteer-supported free office suite for desktop productivity, will be officially announced on 21st August, and available from **www.libreoffice.org/download**.

New features have been developed by 148 contributors: 61% of code commits are from the 53 developers employed by four companies on The Document Foundation's advisory board – Collabora, Red Hat, Hypra and Allotropia – or other organisations, 15% are from seven developers at TDF, and the remaining 24% are from 89 individual volunteers.

Another 202 volunteers – representing hundreds of other people providing translations – have committed localisations in 160 languages. *LibreOffice 7.6 Community* will be available in 120 language versions, more than any other free or proprietary software, and as such could be used by over 5.4 billion people worldwide in their native language. In addition, over 2.3 billion people speak one of those 120 languages as their second language.

If you don't need the latest features, and prefer a version that has undergone more testing and bug fixing, you should install *LibreOffice 7.5.5 Community*, which includes some months of back-ported fixes.

OPINION
JUST BUGGIN'

Jon Masters is a kernel hacker who's been involved with Linux for over 22 years, and works on energy-efficient ARM servers.

"I love to geek out about the nuances of obscure CPU behaviours, including the occasional security vulnerability, in particular those impacting speculative execution hardware. These gathered widespread interest and attention in 2018 with Spectre and Meltdown, silicon bugs that allowed untrusted users to extract information they should not have access to by taking advantage of hidden bread crumbs left behind in the course of running programs and kernel code.

Modern microprocessors are thousands of times faster than the memory and IO to which they are connected. If they had to wait for every access to memory, our modern age would be impossible. Instead, they speculate about the state of a program and its data. If they get it right, performance increases. If they get it wrong, the bad guesses must be cancelled and their effects thrown away. But bugs can result in some of the effects of speculation to become visible to attackers.

Zenbleed seems to be a bit different. Instead of a side channel, data can be speculatively leaked using x86 vector registers. The fix is the same as before: new microcode updates. Read more at **https://lock.cmpxchg8b.com/zenbleed.html**."

Kernel Watch

Jon Masters keeps up with all the latest happenings in the Linux kernel, so you don't have to.

Linus Torvalds announced additional Release Candidate kernels for Linux 6.5, noting that the "release cycle continues to look entirely normal. In fact, it's so normal that we have hit on a very particular (and peculiar) pattern with the rc4 releases: we have had exactly 328 non-merge commits in rc4 in 6.2, 6.3 and now 6.5. Weird coincidence." Thorsten Leemhuis continues to post his reports on regressions, including some keyboard problems affecting Ryzen systems that were supposed to be fixed but apparently are not quite fixed yet.

> "It's going to take some time to remove something so deeply ingrained, but Christoph's latest patches allow a kernel to be built that doesn't use buffer_heads."

Various kernel subsystem teams are discussing their plans for the Linux Plumbers Conference (13th-15th November, Richmond, Virginia, USA). It is not too late to register (either in person, or virtually) or to submit a talk at **https://lpc.events**.

Building without buffer_heads

Christoph Hellwig (known for his contributions to Linux block and filesystem code, among many other areas) has been working on a means to deprecate and remove the buffer_head structure from Linux kernels. As noted in the kernel source (**include/linux/buffer_head.h:struct buffer_head**): "Historically, a buffer_head was used to map a single [disk] block within a page [unit used for tracking memory], and of course as the unit of I/O through the filesystem and block layers. Nowadays the basic I/O unit is the bio, and buffer_heads are used for extracting block mappings (via a get_block_t call), for tracking state within a page (via a page_mapping) and for wrapping bio submission for backward compatibility reasons."

Put another way, they're a very legacy data structure dating back to the earliest kernels, originally for tracking (1,024-byte) blocks of data on disk. Over the years, Linux has added a page cache (structure tracking file content rather than disk blocks) and a bio structure for tracking disk I/O operations in flight but the fundamental buffer_head structure has persisted, with some impact on performance and kernel maintainability. It's going to take some time to remove something so deeply ingrained, but Christoph's latest patches allow a kernel to be built that doesn't use buffer_heads (at the cost of losing many filesystems; these need to be converted before the actual structure can be removed fully). LXF

» ONGOING DEVELOPMENT...

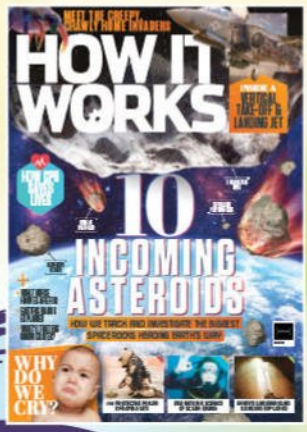
John Ogness posted Wire Up Nbcon Consoles, which aims to refactor how the kernel's printk console logging system works to use the nbcon (non-BKL or Big Kernel Lock) consoles. The idea is to build upon this patch with additional patches that will add threaded printing, atomic printing regions and other goodies that should improve console performance impact on systems. It may sound mundane but in fact this can be a real problem on large servers.

Artem S Tashkinov expressed dislike about how firmware loading is handled: "Many modules don't report which firmware files are getting loaded" and often what is reported does not include the version, date or size of these files. Sam James agreed, and added that Gentoo has been applying a downstream patch "for a little while" to fix this and posted it upstream at some point.

We mentioned the Lichee Pi previously as a low-cost RISC-V development board, similar to a Pi but with a RISC-V chip from T-Head (TH1520). In response to one person attempting to boot an upstream kernel, it was noted that T-Head's OpenSBI (SBI is the firmware interface required to boot and low-level manage RISC-V systems) is several years old. Such issues will probably continue to impede running upstream kernels. With the addition of the BeagleV Ahead board (also using a TH1520) this month, it would be great if such firmware issues were properly resolved.

Summer Savings

SIZZLING HOT SUBSCRIPTIONS



POCHETTINO • PETIT • PARMA IN THE '90S
FourFourTwo
THE 50 BEST
PREMIER LEAGUE PLAYERS
CHARLTON 4-4 SUNDERLAND • HYPNOTISM • CARBONE

OVERSEAS OFFERS ALSO AVAILABLE

PLUS OFFERS ON BINDERS, GUIDES & SPECIALS

See the entire range online at

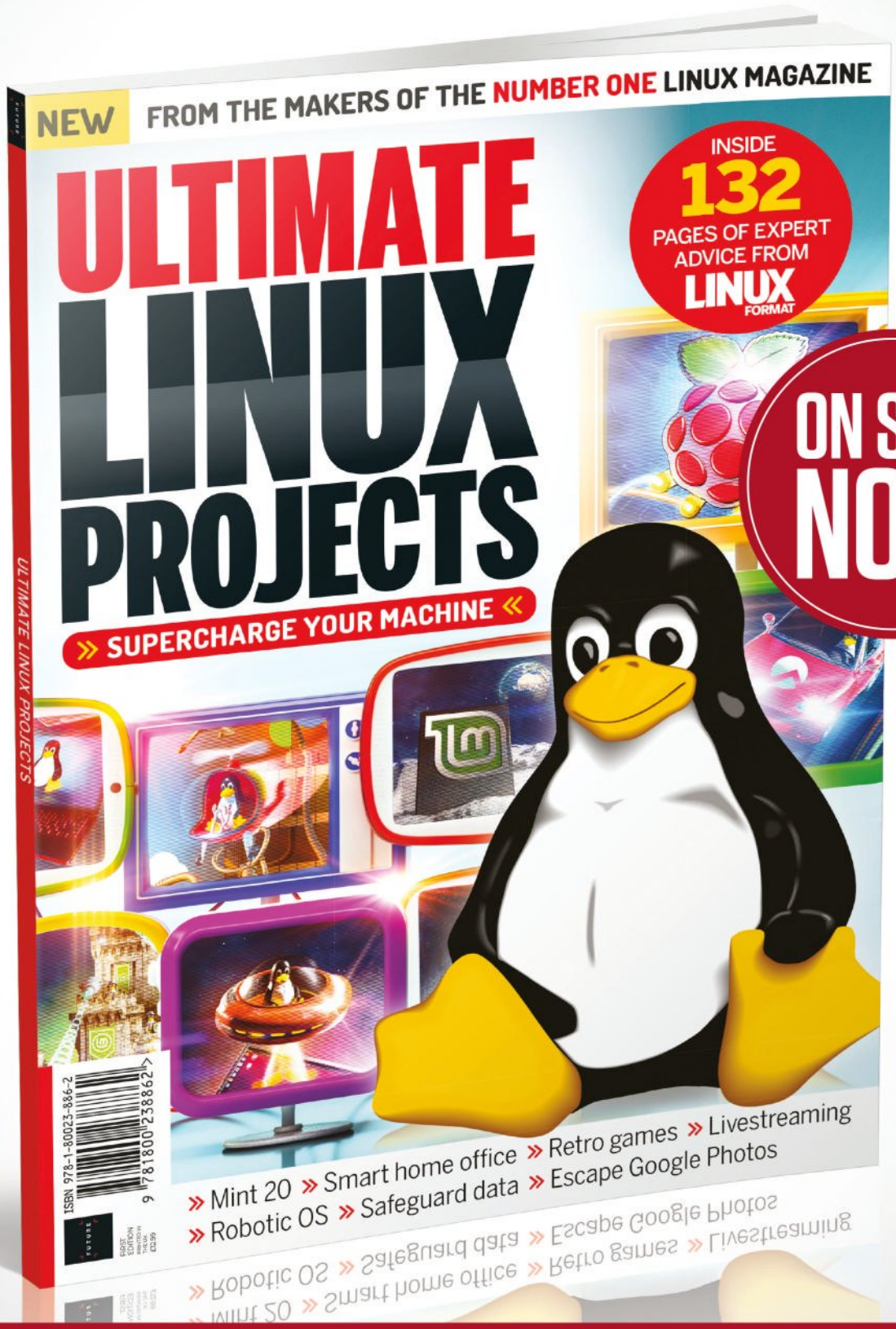
www.magazinesdirect.com/summer23

or phone **0330 333 1113**

Offer closes 30th September 2023. Offer open to new subscribers only. Direct Debit offer is available to UK subscribers only. This price is guaranteed for the first 12 months and we will notify you in advance of any price changes. Please allow up to six weeks for delivery of your first subscription issue (up to eight weeks overseas). The full subscription rate is for 12 months and includes postage and packaging. If the magazine ordered changes frequency per annum, we will honour the number of issues paid for, not the term of the subscription.
For full terms and conditions, visit www.magazinesdirect.com/terms. For enquiries please call: +44 (0) 330 333 1113. Lines are open Monday-Friday 8.30 - 7pm and Saturday 10am-3pm UK Time or e-mail: help@magazinesdirect.com. Calls to 0330 numbers will be charged at no more than a national landline call, and may be included in your phone provider's call bundle.

EXPLORE THE POSSIBILITIES THAT LINUX HAS TO OFFER

From open-source software to coding masterclasses and Raspberry Pi projects, get the most from your machine with this exciting array of expert tutorials, guides and advice from the minds behind Linux Format magazine.


NEW
FROM THE MAKERS OF THE NUMBER ONE LINUX MAGAZINE
ULTIMATE LINUX PROJECTS
INSIDE 132 PAGES OF EXPERT ADVICE FROM LINUX FORMAT
» SUPERCHARGE YOUR MACHINE «
» Mint 20 » Smart home office » Retro games » Livestreaming
» Robotic OS » Safeguard data » Escape Google Photos
ISBN 978-1-80023-886-2
ULTIMATE LINUX PROJECTS

ON SALE NOW

FUTURE

Ordering is easy. Go online at:

magazinesdirect.com

Or get it from selected supermarkets & newsagents

Answers

Got a burning question about open source or the kernel? Whatever your level, email it to **answers@linuxformat.com**

Neil Bothwick likes to read Linux config files at bedtime.

Q Sliding away

The Windows *Photos* program has a good option for amateur photographers. I can make a 'video' of my photos. The photos can move in a good number of ways and I can add background music that adapts automatically to the length of the video. How do I do the same thing in a Linux program? I am afraid you may answer that it is possible, but not nearly as easily.

Bob Abspoel

A What you are looking for is a way to create a slideshow from your photos. There are a few options available to you. They do not have the exact same feature set of *Microsoft Photos,* but they make up for what they miss out on in other areas. The first is *DigiKam,* the photo manager for KDE. Although it is a KDE program, it can be used with any desktop environment and should be in your distro's package repositories. Among its wealth of features is the ability to create a slideshow from an album or selected photos. A range of transitions are available and other factors such as timing and final resolution can all be set. The main omission from your feature list is the ability to add a soundtrack, but that can be done afterwards by loading the video into a video editor, such as *OpenShot.*

Another option is *ffDiaporama* (**https://ffdiaporama.tuxfamily.org**), a program designed to create slideshows from photos. The program is quite old, although there does seem to be some development going on. Old is not always bad, though – once a program does what is asked of it, further feature development could be considered bloat. The interface is easy to use and the results are good quality. However, it did seem a little unstable running it here, but that may be down to other factors on this system – it's a bit of an overloaded workhorse.

The one closest to what you describe may be *Imagination* (**https://imagination.sourceforge.net**). This is a GTK program that does most of what you describe. With almost 70 transitions available, control over timing of slides (both individually and globally) and the options to add an audio track, *Imagination* should be able to achieve what you want.

FfDiaporama is one of a number of programs that can be used to create slideshows in Linux.

Q Historical one-liner

I have found this code, which I think would be very useful if it worked. It looks as though it did long ago (2015?) but I can't make it work. Unless I'm doing something wrong… It should put the filename at the bottom-left of a number of JPEG images.

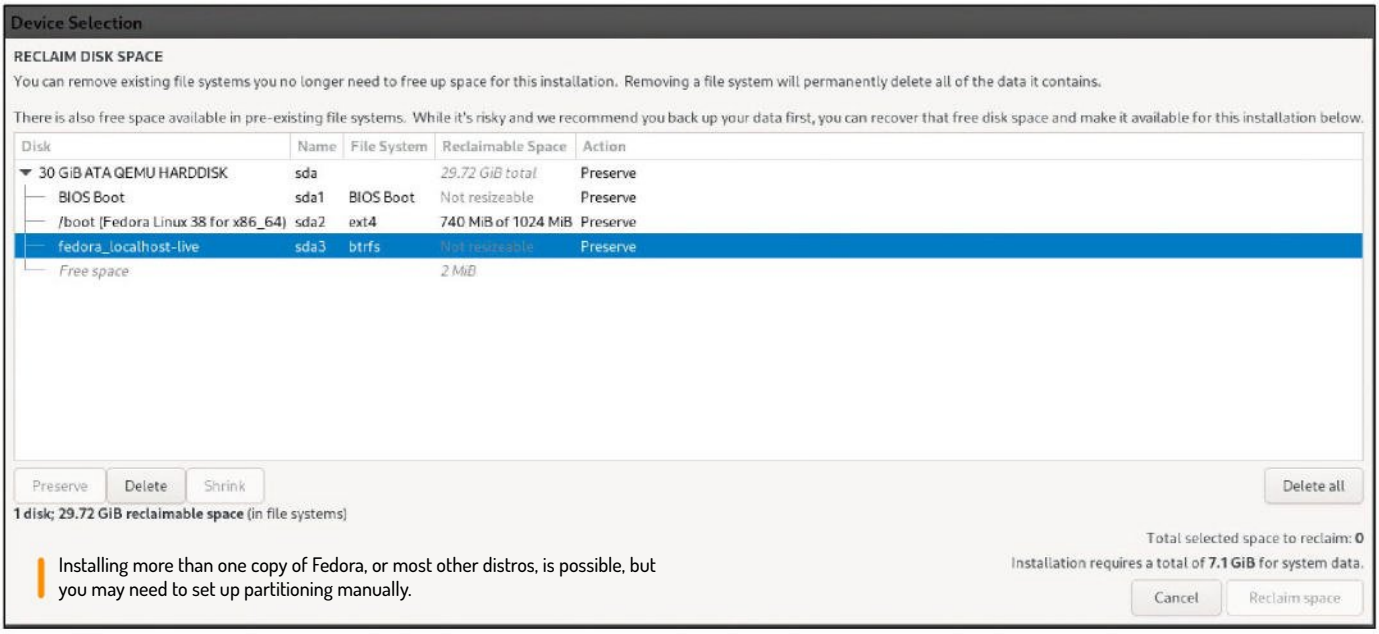
```
for i in *.[Jj][Pp][Gg]; do
  convert "$i" label:"${i%.*}" -gravity East
-append ~/tmp/"$i"
done
```

Dave Fisher

A The loop part looks fine, so try running the `convert` command on a single file until you get it working as you want. There are a few issues here. Firstly, the **-label** (not **label**) option does not draw text on the image; it adds it as a label to the image's metadata. To overlay text on an image, you can use the **-draw** option, although for a straightforward operation like this, the **-annotate** option would be easier. However, that may not work too well either. Potential issues include the text being too small – the size you need depends on the size of the image you are labelling. Also, it can be hard to see the text because you do not know in advance which colour will be at the point you add the text; this is especially true when labelling photos. It is possible to deal with this by adding a shaded box behind the text, called an undercolor in *ImageMagick:*

```
$ convert pic.jpg -fill white -undercolor
'#00000080' -stroke white -pointsize 150
-gravity SouthWest -annotate +0+0 " pic.
jpg " ~/tmp/pic.jpg
```

The **-undercolor** options create a mid-grey shadowed box behind the text. The **-fill** and **-stroke** options set the colour of the text fill and boundary respectively – you usually want these to be the same unless you are going for a fancy effect. The effect of **-pointsize** is pretty obvious and should be adjusted to suit the resolution of your images and how prominent you want the caption. You have already used **-gravity** – it is an easy way to set position, but you need **SouthWest**


Device Selection
RECLAIM DISK SPACE
You can remove existing file systems you no longer need to free up space for this installation. Removing a file system will permanently delete all of the data it contains.
There is also free space available in pre-existing file systems. While it's risky and we recommend you back up your data first, you can recover that free disk space and make it available for this installation below.
Disk
Name
File System
Reclaimable Space
Action
30 GiB ATA QEMU HARDDISK
sda
29.72 GiB total
Preserve
BIOS Boot
sda1
BIOS Boot
Not resizeable
Preserve
/boot (Fedora Linux 38 for x86_64)
sda2
ext4
740 MiB of 1024 MiB
Preserve
fedora_localhost-live
sda3
btrfs
Not resizeable
Preserve
Free space
2 MiB
Preserve
Delete
Shrink
Delete all
1 disk; 29.72 GiB reclaimable space (in file systems)
Installing more than one copy of Fedora, or most other distros, is possible, but you may need to set up partitioning manually.
Total selected space to reclaim: 0
Installation requires a total of 7.1 GiB for system data.
Cancel
Reclaim space

for bottom-left. Get a full list of the **gravity** options in your version of *Convert* with:

```
$ convert -list gravity
```

For **-annotate**, you give an offset – **0,0** here – and the text to add. It helps to add a space at each end of the text to make the bounding box large enough to keep it readable. Once you have this working as you want for a single file, you can plug it back into the loop:

```
$ for i in *.[Jj][Pp][Gg]; do
  convert "$i" -undercolor '#00000080'
-fill white -stroke white -pointsize 250
-gravity East -annotate +0+0 "${i%.*}"
-gravity East -append ~/tmp/"$i"
done
```

Q 2 x 38 = ?

I have a new laptop and have installed Fedora Workstation 38 on it to serve as my work partition. I'd like to set up a second partition on the same SSD to install Fedora KDE so I can use it for play, but I would like it to be set up as a separate partition.

I'll explain why: I love the UI and UX of both Gnome and KDE, the HIG and Kirigami are really nicely polished and I'd like to experience each of them in their entirety (I don't have any custom shell or window themes on Gnome, nor do I plan to get any on KDE).

My question is, can I install Fedora Workstation and KDE on different partitions on the same disk? I typically follow a custom install, with a shared **/boot/efi**; and a **/boot**, **/root** and **/home** for each distro I install. When I'm doing a clean install, I just format **/boot** and **/root** and use a new username on **/home** to carry data over – I use external drives mostly for my files.

Someone I know suggested that I use containers for Gnome and KDE to coexist on one partition but frankly that's beyond what I'm comfortable with, so I was hoping this would work. If it won't, please provide an alternative – it would be appreciated!

Siddharth Sundaresan

A It is possible and quite easy to do what you want. Almost all distro installers support dual-boot installations, either Windows and Linux, or two (or more) Linux distros. The fact that both distros are Fedora here is largely irrelevant. All you would normally do is install the first distro, then run the installer again and tell it to free up space and use that for a separate install. However, Fedora defaults to Btrfs for the system partition and can't shrink it in the installer, so you need to use the manual partitioning option when installing the first instance of Fedora to make sure you leave enough space for the second. Or you can resize with a separate partitioning tool, such as *GParted,* after the first installation. However, this approach is unnecessarily complicated. It uses more disk space and means you have two distros to keep up to date. It also means you have to reboot to switch between your two environments.

A simpler approach is to install both desktops on a single Fedora system. The login screen enables you to pick which desktop you want to use when logging in, so switching between the two environments is as simple as logging out and back in. As these two systems are also for separate use – work and play – as well as different desktops, you could further

» A QUICK REFERENCE TO... NCDU

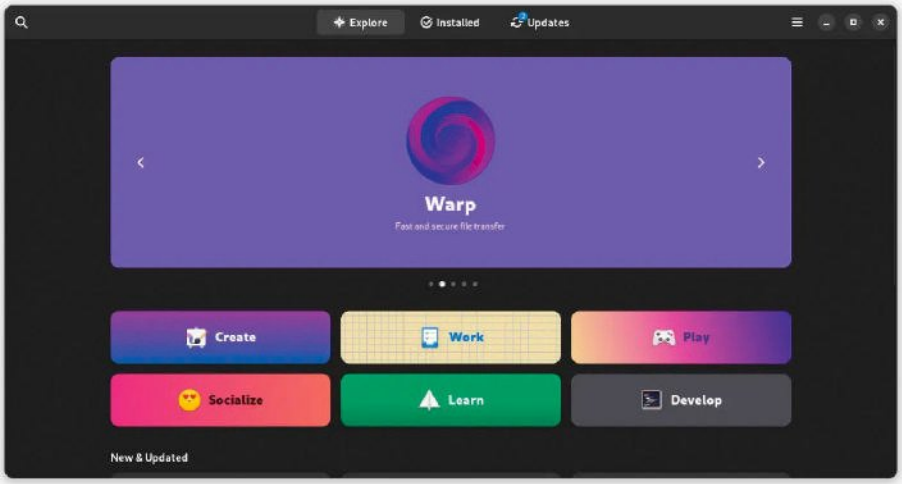
Disk space is one of those things you always think you have plenty of – until you run out. Sometimes disk space can disappear for no obvious reason and you need to be able to find what is using it. The *du* (disk usage) program is the standard tool for this, but it is a bit fiddly to use in this situation. Enter *ncdu,* a curses terminal program that displays the disk usage of a directory, sorted by size, and enables you to drill down through the directories until you find the culprit, such as a runaway log file. It also enables you to delete files or directories, so use with care.

To use *ncdu,* simply pass it the path you want to examine, although it is often worth adding the **-x** flag to keep within the current filesystem. This is especially true if checking / because you don't want it examining network or virtual filesystems:

```
$ sudo ncdu -x /
```

If you are looking at the root, or any other system, filesystem, you must run this with `sudo` or you get permission errors and no useful information.

There are graphical alternatives, such as Gnome's *Disk Usage Analyser* and KDE's *Filelight,* but the beauty of *ncdu* is that it gives as much information in any environment that can run a shell; especially useful if a full disk is preventing your desktop from loading or you are troubleshooting a remote system over SSH.



With the Vanilla OS package management system, each user has to manage packages for themselves.

separate them by creating different users for these two activities. Then you don't even need to log out, just use the Switch User option from the logout menu to switch to a desktop for the other user.

The login manager will remember the desktop you last selected for each user, so this also takes care of switching desktops as well as home directories, to keep your work and play files separate. You could set up a shared area for files that you want both users to have access to. Linux is a multi-user system by design, so having separate users each with their own environments is not only possible, but it is expected.

You are right to discount the use of containers. While this is slightly simpler to work with than separate distros – there is no need to reboot to switch – it is more complicated to set up.

Q Pause for thought

I know how to suspend a shell process with Ctrl+Z, and then use `fg` or `bg` to resume it, but how do I do this for a program that is not attached to a shell, or one that I have already backgrounded with `bg`? I find that when I am doing something processor-intensive, such as transcoding a video, my computer becomes unresponsive, video playback is jerky and so on. I don't want to abort the running process, just pause it and resume it later. I have tried using *Nice*, but it doesn't always help.
Ethan Slater

A You can do this with the *Kill* program. This is usually thought of as being used to terminate processes, but it can also suspend and restart them. First you need the process ID (PID) of the program you want to suspend. You could get this with *Pgrep*, but as this is a program that is hogging the CPU, you can easily spot it in *top*. Once you have the PID, you can stop it running with:

```
$ kill -STOP PID
```

This stops the program running but does not kill it. When you are ready, you can resume the program with:

```
$ kill -CONT PID
```

However, things may not be quite that simple because many programs are multithreaded, spawning other processes. In your video transcoding example, you may see several instances of *Ffmpeg* running, one for each CPU core. You need to send a **STOP** signal to each of these as well as the parent process if you want an immediate result. Sending **STOP** to only the parent process leaves the child processes running until they exit, at which point the parent process doesn't fire up any more. If this is the case, keep track of the PIDs to which you send the **STOP** signal and send them each a **CONT**. Your shell history is your friend here if you didn't take notes. I prefer to **CONT** the processes in the opposite order to which I **STOP**ped them, but that isn't necessary.

You can send the signal from within *top*. Press `k` to be prompted for the PID; the default is the process at the top of the list, which is probably the one you want. Then give the signal to send – **STOP**. *Kill* and *top* also accept numeric signals: **CONT** is **18** and **STOP** is **19**. For the sake of typing two characters, it is usually preferable to stick with the readable names.

Q Users in Vanilla OS

I replaced Linux Mint with Vanilla OS. Under my account, I installed *LibreOffice* and the software store's repository is full of apps. I then added another user. When I logged into their account, *LibreOffice* was missing from within their list of available apps, so I had to install a second copy of the office suite from Flathub. Additionally, the software store was nearly void of apps. Why are the apps I install not available to other users and why are other users' software stores nearly empty? Is Vanilla's containerisation the cause? If so, can I make the apps I install in my account available to all users and how can I have a full list of apps in all users' software stores?
Kevin McMillan

A This is down to the immutable design of Vanilla OS. The root filesystem is small and can't be written to, in order to protect the OS against malware or any other form of corruption. Because of this, everything that is not essential for running the system has to be installed at a user level, using containers. This also means that each user's applications are kept separate, so they can't be corrupted by another user's actions. Yes, it is an inefficient use of disk space, but that is plentiful these days and most programs use far more space for user data than the apps themselves. It is a trade-off for the advantages that Vanilla's way of working gives – one for each administrator to decide whether it is worthwhile.

You can make sure you have the latest list of packages in your store by running the following as your user:

```
$ apx update
```

Apx is the command-line package manager; this command is the equivalent of `apt update` on Debian/Ubuntu, but run as a user. To make sure all users have the latest available, you could create a *cron* task for each user to run `apx update` at regular intervals, say daily or weekly. LXF

GET HELP NOW!

We'd love to try to answer any questions you send to **answers@linuxformat.com**, no matter what the level. We've all been stuck before, so don't be shy. However, we're only human (Kath is constantly having to wake Neil up), so it's important that you include as much information as you can. If something works on one distro but not another, tell us. If you get an error message, then please tell us the exact message and precisely what you did to invoke it.

If you have, or suspect, a hardware problem, let us know about the hardware. Consider installing *hardinfo* or *lshw*. These programs list the hardware on your machine, so send us their output. If you're unwilling, or unable, to install these, run the following commands in a root terminal and send us the **system.txt** file, too.

```
uname -a > system.txt
lspci >> system.txt
lspci -vv >> system.txt
```

Subscriptions: for magazine issues email help@mymagazine.co.uk

Mailserver

WRITE TO US

Do you have a burning Linux-related issue that you want to discuss? Write to us at *Linux Format*, Future Publishing, Quay House, The Ambury, Bath, BA1 1UA or email **letters@linuxformat.com**.

Wrong again #157

It has been said that one can tell what language a programmer used previously by looking at their first *Mathematica* program. Similarly, it is common to see Python programs rated on how 'pythonic' they are.

In a Lisp tutorial, one wants to see Lisp used in the way it is intended to be used, rather than examples that are direct translations from a language such as Basic. If you have learnt Basic, the best advice is to forget all about it, so I was disappointed to see Basic mentioned in a Lisp article together with program examples written in a Basic style.

Lisp is primarily a functional language; purely functional languages have some drawbacks, so Lisp includes some non-functional features, but these are for use in special cases, so should not be included in a first tutorial. Lisp has an REPL (read, evaluate, print loop), which means results are automatically printed, so any mention of the `print` command is an unnecessary distraction. In Lisp, variables are best used only in special cases, so the `setq` command shouldn't be mentioned in a first tutorial either.

It is a mistake to suggest that Lisp is "quirky" (as your 'expert' does). A quirk implies something unpredictable or unaccountable, but Lisp is the way it is for good reasons.

An example to show that besides integers, floating point numbers and strings, Lisp functions can return both lists and functions would go some way to explaining why many of the cognoscenti consider Lisp the best available programming language.

A first tutorial should conclude by providing pointers to further information, which in this case should include **www.paulgraham.com/lisp.html**. I hope you'll try to make sure that future articles will bear all this in mind. I've talked about Lisp, but much of what I've said would apply to articles on other programming languages, too.

John Payne

Sorry, but even the Lisp logo is a bit quirky!

Neil says...

Thanks for the feedback, I do take it on board. We tend to get the odd email from people who know way more about subjects than I or the writers do. I think people's take on how subjects should be approached may not appreciate the magazine's level – though we shouldn't be making mistakes, of course. We're largely here to introduce subjects in a fun manner and not any formal or academic style, and I don't think Mike would ever describe himself as an expert in Lisp, which really is quite quirky from a lay person's perspective, which was always the point of these articles.

Going deGoogled

I am considering buying a phone that is already deGoogled, and also converting an Android phone I bought several years ago to Ubuntu Touch.

I recently saw material about the Murena phone in an article on how to choose a smartphone on the French organisation UFC-QueChoisir's website **www.quechoisir.org** (the equivalent of the UK's Which?).

I myself am more attracted by the phones in the Volla range. The latest model offers its own deGoogled OS, and the option of also downloading Ubuntu Touch as a second OS. Having the option of switching to and from the Ubuntu Touch OS enables the use of the phone as a computer when connected to a keyboard, mouse and monitor.

As far as I can see, the options for saying goodbye to Google include at least eight options:

A. New phones

1. Volla, as above
2. Murena
3. Fairphone
4. FX Technology, the Pro1 X phone,
5. Pine64, for phone and tablets
6. Purism, for the Librem 5 phone

B. Converting phones

7. Rob Braxman, for deGoogled phones
8. UBports

This list shows that there are many ways to get away from Google's control over users' phones, data and privacy.

LXF readers will be interested to read your guidance on whether and how to use F-Droid, MicroG and Aurora, and about apps to replace those that can't be used without surveillance by Google, Apple, Microsoft, Huawei, Facebook and so on. These types of questions are bound to arise in the minds of potential users.

Victor Leser

Neil says...

Hopefully, this month's review of the Murena phone – which is based on the Fairphone 4 – should go some

Helpdex

OH&S SIGNS FOR TECHNOLOGY-BASED WORKPLACES

The latest privacy-protecting phone is reviewed in this very issue!

way to helping answer your questions. We'll be looking at Murena down the line, too. It is a complex area and it's not helped by the fact that some people just want a straight Android replacement while others are looking for the unicorn of full Linux on mobile devices.

Retro Format

I'm using a Raspberry Pi 4 8GB with Raspberry Pi OS on a 64GB SD card, all accessed via *RealVNC,* as the RDP didn't seem stable. I'm not planning on using the Pi for emulation itself but rather for hosting lobbies for my friends. I installed the *Yuzu* emulator via Flatpak, which seems good when it works. I then tried *Citra* but couldn't get it to work with Flatpak or AppImage – even trying an executable, the Snap says it's not ARM64-compatible.

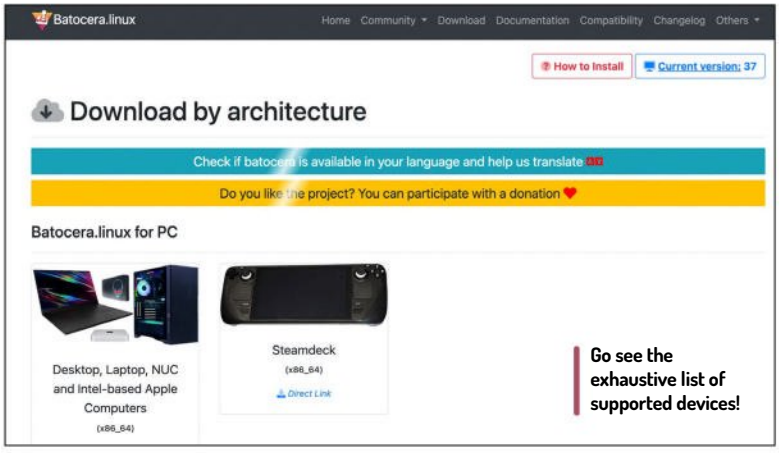
Basically, I'd like to try an N64 emulator with the Aqz plugin, and the PSX emulator using Netplay, but a *Docker* setup also seems an interesting option – I've no idea what that would entail, though.

Richard Smart

Neil says...

Congratulations for embarking on a Raspberry Pi and Linux emulation odyssey. It's a great platform to try emulation on, though it sounds like you're taking a more complicated approach than perhaps you have to. *Citra* looks like it's x86 only, so ARM-based Pis aren't supported, which is the error the Snap was giving you. Frankly, if the Pi isn't supported, you're wasting your time trying to compile this.

If you want a simple emulation solution, you should take a look at Batocera.linux from **https://batocera.org** – it offers not only Raspberry Pi images but also x86 images, so you can use your desktop or Steam Deck among a host of other devices. LXF



LETTER OF THE MONTH

Thoughts on Fortran

I've read consistently that Fortran remains the fastest-executing language, one reason why it's dominant in some aspects of high-end computation to this day (not widely known among **LXF** readership, I imagine). Faster even than C, even though both are compiled, and C is commonly thought of being closer to the metal.

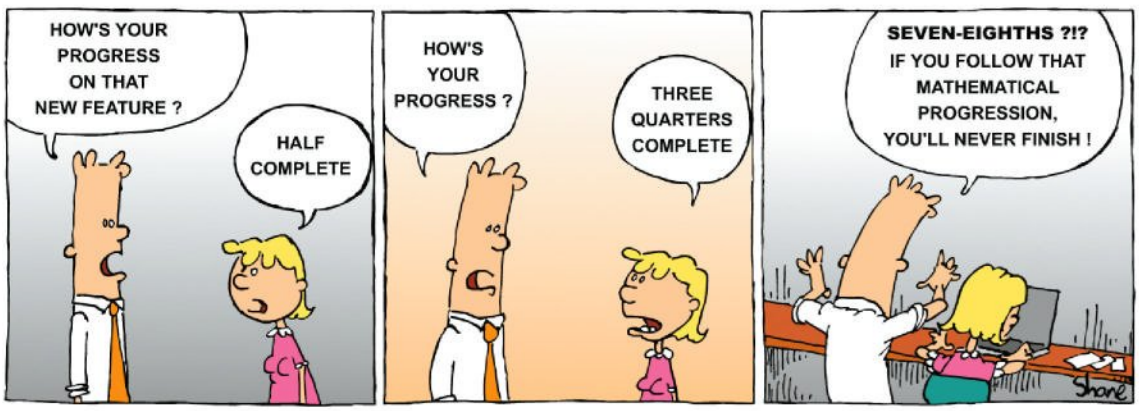
Fortran is sometimes deployed as function calls in other languages, such as MATLAB. C is used the same way. How many readers know that MATLAB was written in Fortran, then rewritten in C? Probably why its vectors start at 1, not 0.

Why not cover why it's so fast? Is it as fast across x86, ARM, Apple, GPUs and CUDA as other language options? You could talk about the criteria and prospects for adding more (is GPL licensing the main barrier, for instance?) and offer predictions about the implications of all this for the future.

Gregory

Neil says...

Covering Fortran is a really good idea. I know it's widely used in scientific circles (my brother, for one, uses it as he's a clever so-and-so with a PhD in physics), so it's probably worth a punt. Not sure we can go into as much detail as you want, but very good suggestions.


HOW'S YOUR PROGRESS ON THAT NEW FEATURE ?
HALF COMPLETE
HOW'S YOUR PROGRESS ?
THREE QUARTERS COMPLETE
SEVEN-EIGHTHS ?!? IF YOU FOLLOW THAT MATHEMATICAL PROGRESSION, YOU'LL NEVER FINISH !
Shane

shane_collinge@yahoo.com

SUBSCRIBE Save money today!

SUBSCRIBE

With every print or bundle subscription, receive an

Alexa AquaBlaster Worth £49.99!

Boompods AquaBlaster Bluetooth 5 portable speakers with Alexa!

- Amazon Alexa approved product.
- Tough multilayered construction with a built-in microphone.
- Call functionality, including play and pause.
- Alexa can be used at a touch by connecting to the *Boompods* app.
- A host of attachment options.
- Fully drop-tested and it's IPX7 waterproof to suit all conditions.

SUBSCRIBE NOW!

www.magazinesdirect.com/LIN/C45D

Call **0330 333 1113** and **quote C45D**

Save money today! **SUBSCRIBE**

FREE DVD! FEDORA 24 & VOYAGER
LINUX FORMAT
65 Pages of tutorials and features
Get into Linux today!
HACK LINUX!
Get under the hood and unlock even more power!
BUILD BOTS | STAY UPDATED
FREE 4.7GB DOWNLOAD
ALL-NEW MINT 19!
MINT POWERED!
64 pages of tutorials & features
FASTER! LIGHTER! BETTER!
DON'T MISS!
Includes 5 years of Linux User & Developer issues

» PLUS: Exclusive access[1] to the *Linux Format* subs area!

Thousands of DRM-free PDF back issues and articles! Get **instant access** back to issue 66 (May 2005) with tutorials, interviews, features and reviews. At **linuxformat.com**

1) Only available to www.magazinesdirect.com subscribers

OUTSIDE THE UK? Turn to **page 65** for more great subscriber deals!

» CHOOSE YOUR PACKAGE!

Four manageable payments over the course of the year, giving you a year's worth of issues in a flexible, affordable way.

ANNUAL PRINT EDITION

Only £16.75
every 3 months
13 issues of *Linux Format* in print over 12 months, **saving 20%**!

ANNUAL PRINT + DIGITAL EDITION

Only £23.00
every 3 months
13 issues of *Linux Format* over 12 months in print and digital, **saving 47%**!

DIGITAL EDITION

Only £13.75
every 3 months
13 issues of digital *Linux Format* over 12 months, **saving 39%**!

*Terms and conditions: **Offer closes 31/10/2023**. Offer open to new UK subscribers only. Pricing is guaranteed for the first 12 months and we will notify you in advance of any price changes. Please allow up to six weeks for delivery of your first subscription issue (up to eight weeks overseas). Your gift will be delivered separately within 60 days after your first payment has cleared. Gifts only available to subscribers on the UK mainland. Gift not available with a digital subscription. Gifts only available in the colour black. The full subscription rate is for 12 months (13 issues) and includes postage and packaging. If the magazine ordered changes frequency per annum, we will honour the number of issues paid for, not the term of the subscription. For full terms and conditions, visit www.magazinesdirect.com/terms. For enquiries please call +44 (0) 330 333 1113. Lines are open Monday-Friday 9am-5pm UK time or email help@magazinesdirect.com. Calls to 0330 numbers will be charged at no more than a national landline call, and may be included in your phone provider's call bundle.

BarraCuda 8TB HDD

Cheap and cheerful – that's not how **Shane Downing** parties.

SPECS

Size: 8TB
Model: ST8000 DM004
Cost per TB: £15.25
Interface: SATA 6Gb/s
Form: 3.5-inch
Tech: SMR
RPM: 5,400
Cache: 256MB
Max rate: 190MB/s
Power: 5.3W
Workload: 55TB/year
MTBF: N/A
Warranty: Two years

One of the least expensive 8TB hard drives you can buy pops it on our list of best hard drives, but this comes at the cost of performance, warranty and reliability. Seagate uses SMR technology to boost capacity up to 8TB at a relatively low price, but this has a negative impact on performance in write workloads and consistency. The 5,400rpm spindle speed doesn't help, either. The BarraCuda does well in read workloads and is adequate for use as media and backup storage; however, the competing 8TB WD Blue that costs around 50% more uses faster CMR technology, making it a better buy for general storage.

Seagate's 7,200rpm FireCuda is the better choice if you want a more robust three-year warranty with data recovery services. The FireCuda is also faster due to its CMR tech and faster spindle speed, earning top billing on our list of best hard drives. However, the extra performance comes at almost twice the cost per TB, leaving the BarraCuda as a somewhat niche product.

The BarraCuda is best when writing large files, then reading back the data. However, writing that data can be problematic due to the trade-offs associated with SMR tech. It is available in an array of capacities from 500GB to 8TB, and all of them use shingled magnetic recording (SMR), which results in severe performance penalties in some types of workloads. The drives also have 256MB of cache at 2TB and above, more than the 128MB on the 8TB CMR WD Blue.

Work to rule

Both the Blue and the BarraCuda come with a short, two-year warranty. The BarraCuda does have a workload rate limit (WRL) that defines how much data you can read and write from the drive before reliability degrades, but it's quite low at 55TB per year during the two-year warranty period. In contrast, the WD Blue doesn't have a workload restriction at all.

The BarraCuda is rated at up to 190MB/s, slightly more than the WD Blue, and it also sips less power than the Blue. This write speed rating may be misleading given that SMR HDDs can encounter performance degradation with certain workloads, such as during sustained data writes.

The BarraCuda is a standard 3.5-inch hard drive, Seagate does offer a 2.5-inch model for laptops up to 5TB. The drives are positioned for 'compute', implying they use SMR tech that enables a lower price per TB versus CMR tech for predictable workloads. Seagate also states the drive uses multi-tier caching (MTC) technology for handling bursty workloads and allowing the drive to consolidate small writes into larger chunks suitable for writing to the shingled data tracks. Like SSDs, SMR HDDs use TRIM to improve performance by reshuffling data in the background.

Using our standard storage benchmarking tool to test file transfer performance with a custom, 50GB data set. We copied 31,227 files of various types, such as pictures, PDFs and videos, to a new folder and then followed up with a reading test of a newly written 6.5GB ZIP file. The BarraCuda copies very slowly, even compared to the Blue, so you should only use it if you don't have a write-heavy use case. Write workloads are far more impacted than reading back the data.

Not all shiny boxes of spinning rust are the same.

The Seagate BarraCuda's read performance is acceptable, even if it peaks lower than the WD Blue, while its write performance suffers at the hands of its SMR tech. The drive lags behind the WD Blue even with larger blocks, but it is especially bad at writing smaller block sizes. As such, it would be best used to write and store larger files and backups, and the trade-offs of selecting an SMR drive are clear in these tests.

The BarraCuda does has very low peak and average power consumption, but it takes longer to finish the workload, so ends up not being particularly efficient. The FireCuda offers much better performance at the same level, while the X300 offers more capacity at the same level. The WD Blue is also a good competitor for average and peak power consumption while using faster CMR technology. LXF

VERDICT

DEVELOPER: Seagate
WEB: www.seagate.com
PRICE: £122

FEATURES	**6/10**	**EASE OF USE**	**9/10**
PERFORMANCE	**4/10**	**VALUE**	**8/10**

It's cheap but suffers from its use of SMR technology. Useful for specific workloads but there are faster CMR options.

» Rating 6/10

GeForce RTX 4060

A good graphics card with the wrong name, ponders **Chris Szewczyk**.

SPECS

GPU: AD107
Arch: Ada Lovelace
Process: TSMC 4N
Clock: 2.4GHz boost
Mem: 8GB GDDR6, 128-bit
CUDA cores: 3,072
Stream cores: 24
RT cores: 24
Tensor cores: 96
ROPs: 48
L2 cache: 24MB
TGP: 115W

Nvidia's RTX 40-series is almost complete. Apart from a probable RTX 4050 and potential RTX 4090 Ti, this GeForce RTX 4060 is shaping up to be one of the final RTX 40-series offerings. As the de facto mass-market Ada Lovelace GPU, it's an important piece of the larger PC gaming puzzle. Should the RTX 4060 be a good performer, it has the potential to drive a wave of upgrades – or, conversely, deter them if it is weak.

Nvidia is positioning the RTX 4060 as a capable 1080p card with excellent power efficiency and support for Nvidia's key technologies, including DLSS 3 with Frame Generation. AMD's Radeon RX 7600 and Intel's Arc A750 are the RTX 4060's logical competitors, but they're not without their flaws. Previous-generation cards remain perfectly viable, too. GPUs such as the AMD RX 6700 XT and Nvidia's own RTX 3060 Ti are still capable gaming cards, and they're available at prices that justifiably keep them in the conversation.

The RTX 4060's compact size, very low idle and load power consumption, AI features and eighth-gen NVENC with AV1 support will surely appeal to users looking for a video card as much as a graphics card. DisplayPort 2.1 support is missing, although in reality 4K and 8K performance is really beyond it.

Perhaps the biggest question, controversy or triviality, depending on how you look at it, is the inclusion of 8GB of VRAM. As is the case with the RTX 4060 Ti 8GB or the RX 7600 8GB, there's no doubt that 8GB will eventually become a bottleneck.

Generation game

The RTX 4060 is built around the AD107 GPU. It's made with a custom TSMC 4N process, which has been tweaked for Nvidia GPUs. The RTX 4060 supports third-generation RT cores with shader execution reordering support, fourth-generation Tensor cores, the eighth-gen NVENC encoder with support for AV1, and, of course, DLSS 3 with Frame Generation capabilities when supported. The card connects to the system via a PCIe 4.0 x8 interface. Again, that's a step back from the full x16 connection offered by the RTX 3060 12GB.

The most impressive feature of the RTX 4060 is its power consumption. At just 115W, with an average gaming power of 98W (and a 7W idle figure), the RTX 4060 is an amazingly efficient graphics card.

In a gaming context, without ray tracing the RTX 4060 tends to slightly outperform the cheaper Radeon 7600 with strong 80-plus frames per second at 1080p and even 1440p managing 60fps in many titles. But don't disregard the Intel Arc 750 or the older Nvidia 3060 that pop in and out of contention. If ray tracing were an option, the RTX 4060 is certainly stronger but performance drops dramatically in half.

Leaning on marketing smoke and mirrors doesn't help as much on Linux.

As for compute performance, the RTX 4060 offers comprehensive speeds, matching the RTX 3060 Ti in *Blender 3.6* performance though falling between the Nvidia RTX 3060 and Nvidia RTX 3060 Ti in *LuxCoreRender* performance. In a CUDA-optimised world, it'll pump out *Stable Diffusion* images three times faster than the Radeon 7600.

The RTX 4060 is an odd GPU for gaming; at £290, it's overpriced against the £260 Radeon 7600 and the older but as capable RTX 3060 with 12GB at £275. For compute loads, it is a powerhouse against any Radeon card – but not the RTX 3060 – and for both gaming and compute, it does its job running coolly and draws less power than the rest. Nvidia still only provides closed-source drivers, which remains a constant irritation, as Radeon just works out of the box. LXF

VERDICT

DEVELOPER: Nvidia
WEB: www.nvidia.com
PRICE: £290

FEATURES	**8/10**	**EASE OF USE**	**9/10**
PERFORMANCE	**8/10**	**VALUE**	**7/10**

We'd name this the RTX 4050 but despite reservations, it would make a solid upgrade at the right price.

» Rating 7/10

Murena Fairphone 4

Fairest of them all, **Jonni Bidwell** is excited by an ethical phone with a privacy-respecting operating system.

SPECS

SoC: Qualcomm Snapdragon 750G, Adreno 619 graphics, 6/8GB RAM, 128/256GB storage
Battery: 3,905mAh
Display: 6.3-inch IPS 1,080x2,340
Rear camera: 2x 48MP
Front camera: 25MP

Netherlands-based Fairphone has made a name for itself with its eponymous line of modular, repair-friendly and, well, fair phones. Besides being made in factories where employees enjoy "good working conditions" (*sounds familiar–Ed*) the phones use ethically sourced and recycled materials, and it places sustainability at the heart of its operation. It has just announced, for example, that the Fairphone 3, launched in 2019, will see Android 13 and a stated aim of two years of support. It continued to support the Fairphone 2 (which launched in 2015 with Android 5 and was upgraded to Android 10 in 2021, see **LXF210** for review) until March 2023, when it received its final update.

The fourth iteration of the Fairphone launched in Europe back in 2021. It's now available in the US, too, but with a new dimension of fairness. To wit, data fairness. Through a partnership with Murena *(see box, right, for the origin story)*, you can buy the Fairphone 4 with the privacy-friendly /e/OS (trying to outdo Pop!_OS in the awkwardly punctuated OS names rubric). The Murena people respect that "your data is your data" (it says so on the box), and as such they don't think it's fair that Google gets so much of it. We should stress this is optional; in Europe at least, you can buy direct from Fairphone with stock Android if you really want.

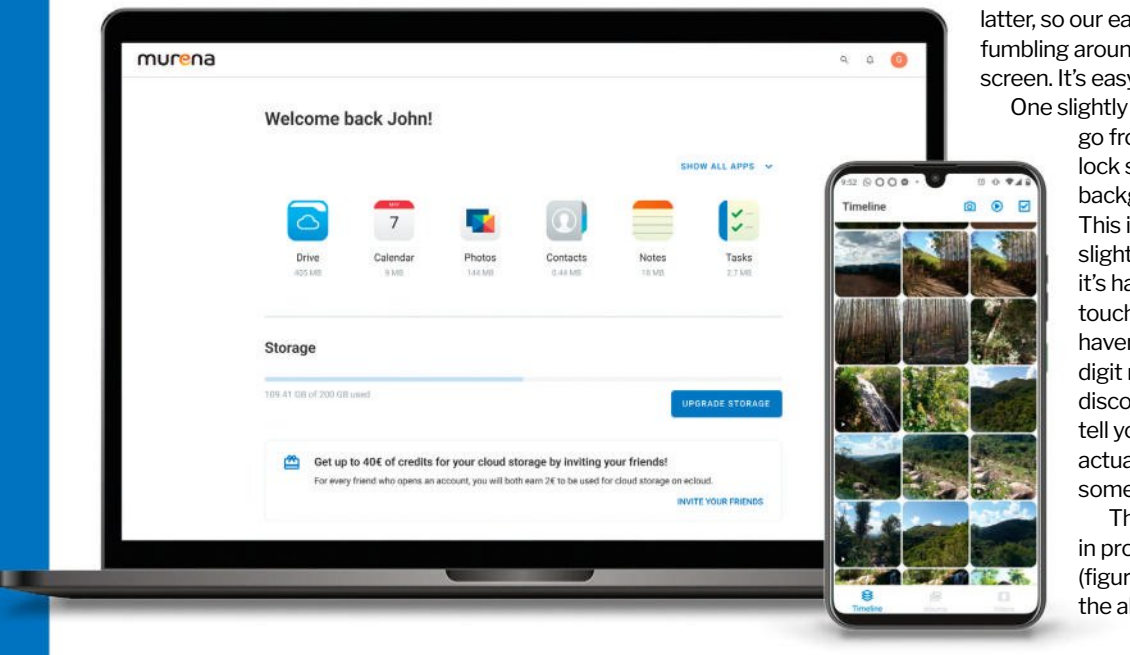
The Fairphone 4 was a mid-range device when it launched, so if you're looking for impressive hardware features today, you're likely to be disappointed. Our digital sibling reviewed it at launch time (see **www.techradar.com/reviews/fairphone-4-review**) and was impressed at the eco-friendliness of it all, but noted the underwhelming camera performance. We're more interested in how the device works with /e/OS, but it's worth restating that you need to tweak camera settings (or get lucky) to get the sort of pictures that come easily to a recent-model iPhone.

The Fairphone is available in two different colours.

Being a modular device makes the Fairphone chunkier than your average phone. With a depth of 10.5mm and weighing in at 225g, you're going to feel this in your pocket. It feels comfortable in the hand, though. And all that mass gives you the sense it can survive the occasional drop. Anyway, let's get to the software. The default /e/OS experience is pretty smooth. It has its own custom launcher which looks (and acts) much more like iOS than Android. We're far more used to the latter, so our early experience included lots of fumbling around trying to get back to the home screen. It's easy to get used to, though.

One slightly jarring experience is when you go from the nice gradients of the lock screen to the austere white background of the PIN entry screen. This is a purely aesthetic point. A slightly more tangible grievance is that it's hard to pick the device up without touching the fingerprint sensor. If you haven't added the fingerprint of the digit making such contact, you feel a disconcerting vibration that's meant to tell you that unlock has failed. What it actually achieved was making us think something was wrong with the phone.

The first sign that you're no longer in proprietary-Kansas any more (figuratively speaking, and apart from the absence of the demands to log



With Murena Cloud, you can sync your photos betwixt PC and mobile with ease.

into a Google account) is that the home screen tells you to set up a weather provider to see the forecast. You can use OpenWeatherMap for this, either with your own API key or one provided by /e/OS (which is rate limited). Once you have a working weather widget, you might notice the home screen also displays Advanced Privacy stats. From here you can disable location services, web trackers and even route all connections over Tor. A graph shows blocked or allowed tracker connections.

The Fairphone 4 is a deGoogled Android device. So, instead of the Play Store, /e/OS has its own App Lounge. The first time you open this you're asked whether you want to see Play Store apps, or just 'PWA and open source ones'. Play Store apps are accessed either by signing in with your Google ID (if you have premium apps tied to your account) or by signing in anonymously. Sign-ins are handled by MicroG (**https://microg.org**), an open source reimplementation of the dreaded Google Play Services (required by many Play Store apps besides Google offerings). MicroG achieves its ends, in part, by stealthily spoofing signatures to allow basic Play Store functionality (it doesn't support purchasing apps or in-app purchases). It's also, since 2020, sponsored by the /e/ Foundation.

Progression sessions

If you've dabbled with deGoogling before, you'll be aware that one of the downsides is app availability. Yes, there are some excellent apps on F-Droid (some are even better than proprietary equivalents), but in many cases, you'll find yourself craving something from the Play Store. On /e/OS, you could just go ahead and install that thing from the Play Store (no need to mess around with the likes of the *Aurora* app). But there's also another way.

We talked about Progressive Web Apps (PWAs) way before they were cool, in **LXF237**. Essentially, they are websites that can run in containers so they look like regular apps. Many popular mobile websites (such as Twitter, Uber and so on) are in fact PWAs, so you can use these services natively with /e/OS. The App Lounge gives you a privacy rating for each app (based on permissions required and trackers included) using data from Exodus. And the Advanced Privacy settings enable you to see which apps are trying to access what trackers and when.

/E/OS even has its own cloud offerings to compete with Google's Calendar, Drive and Gmail. Ardent data sovereignty types might already run their own instantiations of these services, but that's not something that an average human can be reasonably expected to do (especially running your own email, *shudder*). Offering these services for free is a bold move, and extra points to team /e/ for providing packages so that people like us can self host them.

In sum, the Fairphone 4 is a middle-aged device and people may baulk at the price tag (and its grainy photos). We're looking forward to the camera upgrade when the Fairphone 4 Plus launches (like the previous generation, components will be individually upgradeable). But Murena became industry leader by bringing deGoogled phones to market, and this new flagship will rightly have the data-conscious reaching for their wallets. LXF

VERDICT

MANUFACTURER: Fairphone/Murena
WEB: https://murena.com
PRICE: £450–£500

FEATURES	**9/10**	**EASE OF USE**	**8/10**
PERFORMANCE	**7/10**	**VALUE**	**7/10**

You can't put a price on privacy but if you could, the /e/OS-powered Fairphone would score better in the value category.

» Rating 8/10

» /E/OS IS EBENEEZER GOOD

In 2017, developer Gaël Duval (who created Mandrake Linux, later Mandriva Linux) announced he was working on a Google-free version of Android. Even then, there were a few options for those wanting a deGoogled phone, most notably CyanogenMod (later LineageOS) and Ubuntu Touch. But these (and the many other options now available) have never suited everyday users. Having to unlock bootloaders and being denied familiar G-apps does not a friendly UX make.

Duval launched an IndieGogo campaign to drive interest for his OS, which he described in an interview with The Register as "something that mum and dad will like to use". Interest was piqued and the project, initially called eelo, forked the LineageOS codebase and got underway in earnest. Eelo, the overarching project (which was now eying industry partnerships), was later renamed Murena, with the OS itself being renamed /e/OS. This culminated in Murena bringing the first /e/OS phone (the Fairphone 3) to market in 2020.

Murena launched its own device (the Murena One) in 2022 and (besides the Fairphone 4) now also offers /e/OS-bearing Pixel 5 (factory refurbished) and Teracube handsets. Today, /e/OS can be installed (easily) on over 200 devices. The /e/ Foundation (a non-profit set up by Duval) provides stewardship to the project. We'll be covering /e/OS in more detail next issue, so stay tuned.

We couldn't help but notice the /e/OS monogram bears a passing resemblance to that of a certain Borg-like outfit.

Murena currently offers four different devices with /e/OS, three of which are pictured.

Peppermint OS

After years of sampling Linux Mint, **Nate Drake** opts for something spicier in the hybrid Peppermint OS, which integrates cloud-based apps.

IN BRIEF

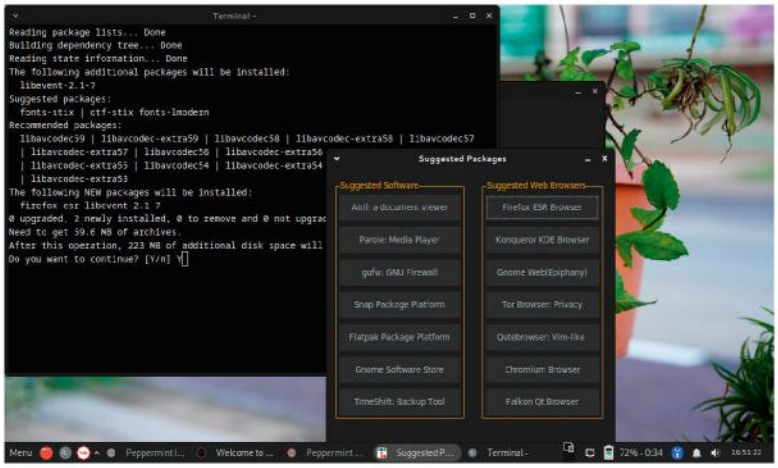
Peppermint OS is visually rich and the *Calamares* installer makes setup a dream. The Welcome Hub also has a good range of suggested software but it has minimal bundled apps.

SPECS

CPU: 1GHz
Mem: 1GB (4GB recommended)
HDD: 10GB (32GB recommended)
Builds: x86_64

Like many great ideas, Peppermint OS was conceived after a night in the pub. If you're wondering about the moniker, one of the lead developers, Kendall Weaver, also works on Linux Mint. The name itself comes from the creators' desire to build on Mint's success but also add web integration, hence the extra zing of 'Peppermint'.

The project has had a mixed history, previously being based on Lubuntu and incorporating elements of the LXDE desktop environment. These days, however, there are two main branches of Peppermint. One is based on the Debian fork (Devuan), while the version that forms the basis of this review is based on the most recent stable release of Debian itself (Debian 12 Bookworm). It uses the lightweight Xfce desktop environment and the inimitable *Thunar* file manager.



Peppermint OS offers a way to install popular software and web browsers, such as Firefox ESR, from Debian's stable repositories.

Although the ISO weighs in at around 1.5GB, on first boot you'll find there are very few pre-installed apps. This is because Peppermint has a hybrid desktop that uses both cloud and local apps. The OS contains *Kumo*, which can be used to launch SSBs (site-specific browsers). This means a dedicated browser window is assigned to specific cloud-based applications. The latest version of *Kumo* comes with an updated GUI and is based on the universal Lua programming language.

This can sound a little daunting but Peppermint makes an effort to hold your hand. If you boot into the live desktop environment, you'll see the Peppermint Hub, which has a list of suggested apps to install. This facilitates you downloading the latest version of certain apps from Debian repositories. However, if you want to install apps from Peppermint's own repos, you need to update the **Sources** file yourself manually.

If you're happy with what's on offer, the latest version of Peppermint has now judiciously divided this into two sections: Suggested Software and Suggested Browsers, presumably to make the layout simpler – we used the latter to install *Firefox ESR*.

You can also launch the *Synaptic* package manager or download and set up *Gnome Software*. The updated Welcome screen is, in fact, one of the features in the latest version of Peppermint.

The most recent iteration of the OS, version 2023-07-01, has also updated its Pep Docs, to help ease new users into the distro's way of working. This is where we learned about **hblock**, a super version of the regular **hosts** file found in Debian and its derivatives. It includes a list of domains that are known for serving ads, tracking scripts and malware, and prevents your cloud-based apps from connecting to them.

The Pep Docs also mention the bundled *XDaily* tool, a command-line system updater/upgrader that, once launched, only requires you to answer Y/N to perform routine tasks such as updates, removing old caches and updating your icon cache. You can also launch this using the handy little blue icon in the system tray.

The help guide also points out the range of different packages the OS supports, though it's down to you to work out which is best for your Peppermint hybrid desktop. Speaking of the desktop, the latest version of Peppermint uses the Marawait theme and Tela icon set. These circular icons are particularly easy to tell apart from the quick-launch bar at the bottom-left.

If you do decide to go ahead and give Peppermint a try, setup is a breeze via the *Calamares* installer. From there, though, you're pretty much on your own if you haven't previously initialised cloud-based apps. Luckily, if you need help, the team maintains a thriving forum, Matrix channel and Mastodon page. LXF

VERDICT

DEVELOPER: Peppermint OS Team
WEB: https://peppermintos.com
LICENCE: GPL

FEATURES	**7/10**	**EASE OF USE**	**7/10**
PERFORMANCE	**8/10**	**DOCUMENTATION**	**7/10**

Peppermint is stunning and very easy to install but the job of setting up SSBs might be too daunting for newcomers.

» Rating 7/10

Fatdog64 Linux 814

Nate Drake decides to look in on Fatdog, a Puppy-based OS that's grown into a very credible canine in its own right.

IN BRIEF

Fatdog is the puppy who grew up. It contains an excellent web browser, plus productivity, media and image-editing tools. It loads in seconds and has an excellent package manager.

SPECS

CPU: 600MHz
Mem: 1GB
HDD: 600MB
Builds: x86_64, ARM

Every dog has its day and for the team at Fatdog, there have been many. The project started out in 2008 as a simple extension of Puppy Linux. Any experienced Linux user will tell you that Puppy's designed to be lightweight and easy to use, but the number of customisation options and requirement to install extra software to have a workable OS make it unsuitable for beginners.

Fatdog (as the name suggests) is the pup grown up. Its developers have placed emphasis on a fatter – that means more bundled apps – version of Puppy, while remaining very fast and efficient.

The OS includes a primer for users coming from Puppy to the more fully-featured Fatdog64.

We couldn't see any better proof of this than when downloading one of the latest versions (814) of Fatdog, as the ISO weighed in at just over 500MB. The OS is also compatible with pretty much any computer using an Intel/AMD CPU released since 2008. On first boot, we chose to load the OS in Throwaway mode, which means into RAM but, like Puppy, you can install to a drive if you prefer. If you do this, Fatdog can store its settings in your existing FAT, Ext2/3/4, NTFS partition using two simple files – no partitioning required.

Monkey business

On first boot, it's clear the team has lived up to its claim of having a desktop ready to use. Incidentally, the desktop environment uses the extremely versatile and lightweight *Openbox*. The official web browser is *SeaMonkey (v2.2.2)*, though we noted the Internet menu also lists *Firefox* and *Chrome*, which can be installed via a mouse click.

There's a host of video and sound editing tools, and we were particularly impressed to see the latest version of *VLC*. Productivity apps include *LibreOffice Writer, Calc* and *Impress*, as well as the *Evince* PDF viewer. Fatdog also includes a range of image-editing tools, including the lightweight *mtPaint* and *GIMP*.

If you want to install extra software, Fatdog packages come in two flavours. The first is TXZ and operates along the same lines as Slackware, in that it contains a single compressed file with all the application data. (The Fatdog team stresses, though, that not all Slackware packages are fully compatible with the OS.) The other package type is SFS (Squash File System), the same as that used by Puppy Linux. It can contain multiple applications and libraries.

While Fatdog has become an independent and powerful distro in its own right, it isn't fully off the leash from its Puppy masters. The OS even comes with a primer (also available online) for Puppy users about how to make the transition.

Fatdog doesn't have its own dedicated forum but relies on a subforum on the Puppy Linux website (**https://forum.puppylinux.com/viewforum.php?f=60**). This is where we discovered that the release of Fatdog 64 814 will be the last in the 800 series. It contains minor improvements over the previous version, such as being based on version 5.19.17 of the Linux kernel. Other updates include the most recent versions of *VLC* and SMB, as well as *Avidemux 2.8.1*. There have also been minor tweaks to the *Bluetooth Manager* and the inclusion of a new *Geany* themes bundle.

The Fatdog team never fails to surprise, as simultaneous to its announcement of the latest stable 800 series came the news that it's releasing an alpha build of Fatdog 900. The team claims to have been using it behind the scenes for several months. It can only be installed, not upgraded to, but improvements over Fatdog 814 include a full system rebuild with brand new package repos. Head to **https://distro.ibiblio.org/fatdog/web/latest.html** to find out more.

VERDICT

DEVELOPER: The Fatdog Team
WEB: http://distro.ibiblio.org/fatdog/web/
LICENCE: GPL v2

FEATURES	**8/10**	**EASE OF USE**	**8/10**
PERFORMANCE	**9/10**	**DOCUMENTATION**	**8/10**

Fatdog lives up to its name, being a derivative of Puppy that just works straight out of the box.

» Rating 8/10

Roundup

Modicia OS » Ubuntu Studio » Fedora Jam » Fedora Design Suite » AV Linux

Michael Reed loves to get creative, so he's all over anything open source that helps with that.

Media-creation distros

Michael Reed examines five distributions, aimed at creative types, that come packed with applications, utilities and plugins.

HOW WE TESTED...

Media creation is a very broad spectrum. The requirements for something such as drawing with a tablet might be very different from those for recording music. For one thing, setting up an art station might be a good use for an older computer, whereas audio recording tends to hit the hardware resources to a greater extent.

Many of the smaller support tools for audio work, such as utilities and plugins, can be quite fiddly to get working. For example, there is often no universally agreed on location for plugins.

Most Linux installations require quite a few tweaks to work efficiently for audio creation and we wanted to make sure that each of these distros was adequately optimised in advance. It's not really fair to say that a distribution is ready to work with audio if, in actual fact, the playback and recording would glitch and stutter even though applications such as *Ardour* could technically run.

Linux has become a creative hotbed, from *Blender* to *Krita* and *Audacity* to *Ardour,* but setting up these complex packages can be non-trivial. Enter media-creation distributions. These are distros that come packed with pre-installed and preconfigured applications and utilities. This software can cover a range of different creative areas, such as graphics, video or music and audio. Although they can be used as standard Linux distributions, most of the ones we're examining here feature all sorts of tweaks that make them particularly suitable for media work. Also, all five of these distributions can function as a live distro, making them even more useful when creative juices are flowing but you have to use someone else's setup.

Ubuntu Studio is, as the name suggests, the official flavour of Ubuntu Linux that's optimised for media creation. Fedora Jam, as the name also suggests, is based on Fedora Linux, and it has an emphasis on music and audio apps. Fedora Design Studio is another Fedora-based respin, but this one has more of a focus on art and design. AV Linux has a broad media-creation premise, and it's based on MX Linux. Modicia OS has flashy looks and some interesting optimisations, and uses Ubuntu (LTS release) as its base.

CREDIT: Getty Images/iStockphoto

Customisations & extra features

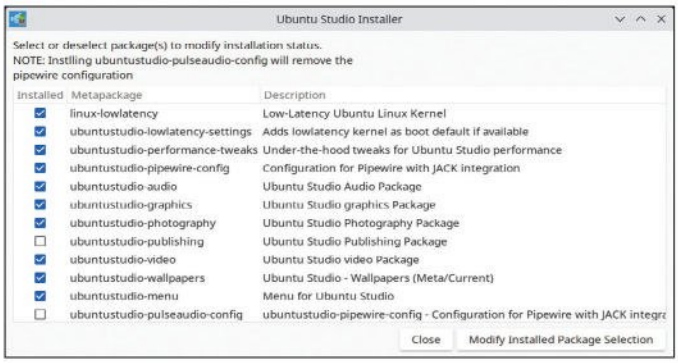
How much does each distro add?

All the distributions, apart from Fedora Design Suite, have the JACK system installed and configured, which is what most Linux audio apps expect. Of those four, only Fedora Jam lacked the customisation of adding the user to the 'audio' group, something that makes audio apps run more smoothly. A slight wrinkle affecting them all is that their base distros now run on PipeWire, a new audio system. This means you may still have to tweak your system for best performance.

AV Linux uses the Liquorix Kernel (**https://liquorix.net**), a special build of the Linux kernel that is tuned for multimedia work in areas such as swap usage, process scheduling and CPU frequency. The compromise is that it gives lower throughput in some situations and consumes more power, something to bear in mind if using it on a laptop. Having a non-standard kernel could throw up some unusual problems in rare cases, but overall, having Liquorix Kernel installed is a boon for multimedia work.

As AV Linux is based on MX Linux, it comes with MX Tools, a set of GUI configuration tools that cover many areas that tend to be left for the user to tweak manually in most distros. These include a tool to show various aspects of system and hardware information, a tool to install media codecs, and a tool to mount iPads and iPhones, and many more. Its inclusion adds to the feeling that AV Linux is something of a tweaker's distro.

Ubuntu Studio uses the low-latency kernel. This is an officially supported variant of the standard kernel, meaning that it's a little closer to a standard setup than one that uses the Liquorix Kernel, and it does offer improved performance for real-time multimedia tasks. Ubuntu Studio is the only one of the distributions that comes with a GUI tool to remove PipeWire if you prefer to revert to the older-style setup until PipeWire becomes more mature.



The Ubuntu Studio Installer lets you quickly add and remove certain customisations for troubleshooting and performance tweaking. You can also remove PipeWire.

Modicia OS runs the standard 'generic' kernel, but the developers have applied some system tweaks in areas such as swap file usage to improve multimedia performance. Usefully, Windows emulation layer WINE is installed and configured.

VERDICT

MODICIA OS	**7/10**	**FEDORA DESIGN SUITE**	**5/10**
UBUNTU STUDIO	**8/10**	**AV LINUX**	**8/10**
FEDORA JAM	**7/10**		

Ubuntu Studio has most of the customisations that you need for a media-creation computer applied in advance.

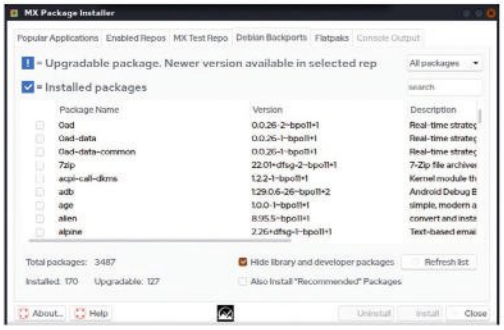
Base distro and package repros

We need a strong foundation with access to the latest software.

Understandably, all of these distributions only offer 64-bit builds. They all have some capability as a live distro. For some of them, the preferred base distro will come down to personal preference and previous Linux experience.

AV Linux is based on MX Linux, which itself is based on the stable branch of Debian. This gives command-line tools that will be largely familiar to users of other Debian derivatives, such as Ubuntu and Mint. It draws its packages from the MX Linux repositories, which is a mixed blessing. In some areas, these repositories are ahead of what Ubuntu Studio has to offer in the way of audio software; in other areas, Ubuntu Studio has the more up-to-date packages.

Fedora Jam and Fedora Design Suite are spins of Fedora Linux, and therefore use the Fedora repositories and Fedora command-line tools, such as the *DNF* command-line package manager. These repositories are, overall, slightly ahead when it comes to offering up-to-date music tools. In addition, both distros have access to Flatpak packages by default.



The MX Package Installer has access to the MX Linux, MX Testing and Debian Backports repos, as well as Flatpak, giving access to some bleeding-edge packages.

Making the issue more complicated, the Debian/Ubuntu-based distributions can add the KXStudio repositories, which are the most up to date repositories of multimedia tools and applications, a big advantage. PPAs are another source of up-to-date software.

Modicia OS is based on Ubuntu LTS, which might be a disadvantage as some cutting-edge media tools, particularly plugins, require bleeding-edge libraries. Ubuntu Studio, on the other hand, offers LTS and current stable release versions.

VERDICT

MODICIA OS	**6/10**	**FEDORA DESIGN SUITE**	**7/10**
UBUNTU STUDIO	**9/10**	**AV LINUX**	**8/10**
FEDORA JAM	**7/10**		

It's a mixed bag. Ubuntu Studio has access to the most up-to-date software. The Fedora spins are also good.

Using the user interface

Creative flow is paramount.

The requirements of a GUI that's used for multimedia aren't massively different from that of a normal desktop. The interface should be quick and easy to use, as you often need to be on the ball when using Linux as the hub of your studio. In the case of distributions that offer a large number of tools, the launcher should be convenient and make use of categories. Clarity in areas such as launch icons is welcomed.

We expect customisation features that are easy to use, because when you're working on creative endeavours, you often need to customise the layout to enable your workflow. To us, that means that you can easily make top-level selections such as changing the overall theme and the font size, as well as adjusting small details or adding features such as extra panels to the desktop. Oh, and we prefer configuration features that are easy to locate and preferably centralised and searchable. Options that can't be found aren't much use.

AV Linux 7/10

The default AV Linux theme is dark, and it makes some use of transparency and other effects, giving flashy looks at the expense of some extra resource usage. Along the top is a large dock with some major media-creation apps, while on the bottom is a smaller taskbar and launcher. On the backdrop, there's a useful status display area showing real-time updates on essential info such as kernel version, memory usage and CPU temperature.

As you might expect from an Xfce desktop, there's great scope for customisation, but the relevant features are spread out across different tools. This is largely down to how Xfce works, and the MX Tweak tools further duplicate the appearance options. We hit another snag when switching to the light theme that resulted in a nearly unreadable terminal.

It's a good-looking tweakable desktop that will appeal to the technically minded.

Fedora Design Suite 6/10

The desktop here is a rather minimalist implementation of the standard Gnome one. When you first log in, you're greeted with a pretty airbrushed backdrop and the standard Gnome status bar. Pressing the super key or clicking in the status bar invokes Gnome's combined launcher and app switcher. This means an extra click or two, as you can't switch tasks without using the full-screen switcher.

From the launcher screen, you can find apps via searching by typing, or you can click on the Show Apps icon. From here, you can browse the installed apps by icon. A snag is that there are few categories and they are thrown in with the app icons.

The minimalist approach might have been designed to offer a distraction-free environment for artists, but having no dock leads to extra steps when launching commonly used applications or switching between them. It's an odd choice.

System efficiency

It's a shame if the distro starts gobbling up resources before you've even started.

For most media-creation work, you need a fairly well-specified computer, and none of these distros is designed for low-end hardware. However, multimedia work tends to push a computer to its limits, so it's still worth considering if a distro is efficient.

AV Linux keeps things light, in typical fashion for an Xfce-based desktop. Fully updated when freshly booted to the desktop, it used only 467MB of RAM and 13GB of disk space.

Fedora Design Suite is a heavyweight, eating 1.5GB of RAM on a fresh boot to its desktop, but it only used 6GB of disk space.

Ubuntu Studio wasn't that far behind and reported 1.1GB of RAM in use on a fresh boot. That amount doesn't break the bank and KDE Plasma isn't designed to be a lightweight desktop.

Fedora Jam is also a KDE Plasma-based distro, and yet it manages to boot to the desktop consuming a little less RAM, at 933MB. Applications themselves tend not to be space hogs, and that must be why it was the lightest of the distros in this area, only using 5.5GB of disk space.

Modicia OS takes up 15GB on the disk and 943MB of RAM upon a fresh boot. It's worth noting that it has some fairly heavy compositing effects enabled, such as windows that wobble and bend when dragged about. For this reason, you might like to disable them to free up CPU/GPU resources unless you're certain that it won't be an issue.

VERDICT

MODICIA OS	**7/10**	**FEDORA DESIGN SUITE**	**6/10**
UBUNTU STUDIO	**7/10**	**AV LINUX**	**9/10**
FEDORA JAM	**8/10**		

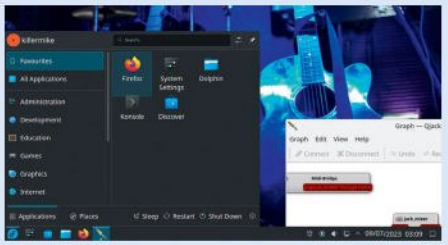
AV Linux used the fewest resources, but Fedora Jam packs in a lot for a fully-featured desktop.

Fedora Jam 8/10

Although it comes from a similar origin, Fedora Jam has a different front-end from Fedora Design Suite as it uses KDE Plasma as the desktop environment. The default layout is fairly conventional, with a combined launcher/task switcher/status bar at the bottom of the screen.

The theme is dark and makes some use of transparency, but as is usual with Plasma, it's a clear layout. Sure enough, it's quick and easy to change to a light version of the basic theme and this theme also looks good. Generally, the customisation options are excellent and easy to access.

The application launcher is searchable. The layout may be age-old, but it is at least easy to browse by categories. Most of the music tools that come with this music-centric distro are in the Multimedia category, and it would have been nice to have seen some custom categories. It's a good, if conventional, Plasma desktop.

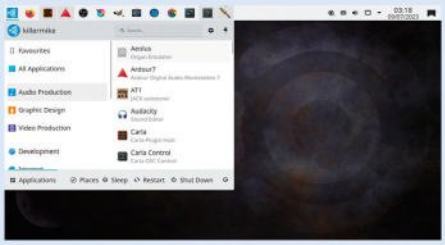


Ubuntu Studio 9/10

The Ubuntu Studio desktop is dark-themed and uses KDE Plasma. The main panel is icon based and sits at the top of the screen. Some media applications and tools have been added to the quick launch panel. This was a bit on the small size, and bumping it up was one of the first changes we made. This aided clarity and put the launch icons within easier reach.

Being KDE Plasma, configuration tools are centralised and searchable. It's dead easy to change to a light theme that is just as clear and attractive as the dark one.

There are separate custom categories for audio, graphics and video, rather than just having everything lumped into the multimedia category. Beyond that, it's a fairly standard KDE desktop, but that's no bad thing. There may be slightly lighter desktops when it comes to resource usage, but as it stands, it's quick and clear, with a searchable launcher.



Modicia OS 7/10

Modicia OS defaults to a dark theme but it gravitates more towards midtones than the grey on black that we're used to. It's actually Cinnamon (a desktop that leverages Gnome 3 tech) providing the desktop, but you'd struggle to recognise it at first because it's highly customised.

Along the bottom of the screen, there is a dock and we liked the fact that it's a decent size by default. As soon as you hover over this dock, you'll be struck by the fact that it stretches and bends. In fact, Modicia OS makes use of extensive compositor effects throughout. If this isn't to your taste or you can't spare the resources, you can disable those effects.

Here, we found that Modicia OS sits in the middle when it comes to ease of configurability because the options are reasonably centralised (and searchable), but they are manipulated from within a variety of configuration applets.

Audio, music and plugins

Showcasing what Linux can do in this realm and saving installation effort.

Fedora Design Suite ships with an app selection that focuses on art, design and publishing, with a few multimedia apps (*Pitivi*, *Audacity)* thrown in. For audio and music work, it's on the same level as a standard Linux distro.

Fedora Jam, Ubuntu Studio and AV Linux offer a very similar overall package, in that they are all packed with music apps and tools. In each case, the application selection is extensive but free of duplication. Each of them has favoured *Ardour* as the DAW.

Sure enough, for each of those distros, the first time we ran *Ardour,* it carried out a scan and discovered around a dozen instrument plugins and around 50 effects plugins. Those three distros didn't offer all of the same things, but we struggled to put any particular one ahead of the others. Each of them would be an excellent starting point for exploring what Linux has to offer for music creation and audio editing. AV Linux differed slightly as it also offered some demo versions of commercial software, such as *DAW Reaper,* and some commercial plugins. Whether you like this probably comes down to personal preference.

Modicia OS is fully set up for audio and music work, but it offers a slimmer selection of applications and plugins. The DAW that it offers is *Qtractor,* a more traditional, MIDI-orientated sequencer. There are some audio effect plugins but we couldn't find any instruments at all. Chances are, you would have to do a bit of installing to get any real music work done.

VERDICT

MODICIA OS	**6/10**	**FEDORA DESIGN SUITE**	**5/10**
UBUNTU STUDIO	**8/10**	**AV LINUX**	**8/10**
FEDORA JAM	**8/10**		

Fedora Jam, Ubuntu Studio and AV Linux offer a similar selection of music applications and plugins.

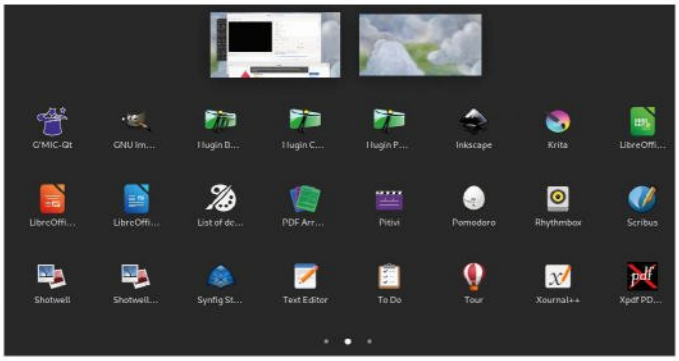
Video and graphics apps

Ready-to-go drawing and painting applications are always welcome.

Fedora Design Suite has an emphasis on some of the best graphics tools that Linux has to offer. *Inkscape, Krita* and *GIMP* are the standouts. It's nice to see diagram-creation application *Dia,* photo editor *Darktable* and the 3D graphics behemoth *Blender,* too. There are a few other smaller art-orientated tools included, but it's a fairly basic collection of applications. The idea might have been to avoid overwhelming the user, but we would have accepted duplication of features if it meant a greater number of tools.

Ubuntu Studio is a strong distro for audio work, but it also offers a good selection of art and graphics applications. *Inkscape, Krita* and *GIMP* are present, among others. On the video side, we have *Kdenlive* and the excellent screen-recording application *OBS Studio.* We gave it an extra point for the custom categories that helped us to find all of this stuff.

AV Linux offers a decent selection of graphics and video applications along with some useful GUI tools in this area. Modicia OS covers the graphics basics, and it adds *RawTherapee* as the photo editor, another good choice.



A page of the apps that Fedora Design Suite has to offer. It covers the basics, along with some extra graphics tools, but we felt the selection was a little conservative.

Fedora Jam offers little beyond the basics for graphics. *KolourPaint* is OK for basic doodles and simple image edits. As *LibreOffice* is installed, the *Draw* module is available as a basic vector drawing program. There's nothing of note when it comes to video editing. All in all, the selection of graphics and video software is on about the same level as a typical distribution.

VERDICT

MODICIA OS	**7/10**	**FEDORA DESIGN SUITE**	**7/10**
UBUNTU STUDIO	**8/10**	**AV LINUX**	**8/10**
FEDORA JAM	**5/10**		

Fedora Design Suite should have pulled ahead here, but most of the other distros did well, too.

Documentation and support

Information on how to get things running and overcome difficulties.

The Modicia OS website looks a bit messy, and the PDF manual is out of date. The development team offers support to users who make a donation. In short, there isn't much free support and you have to seek documentation for the individual components rather than for Modicia OS itself.

As you might expect for an official Ubuntu flavour, Ubuntu Studio is extremely well documented, with extensive support options. For example, the documentation section has mini manuals on subjects such as using common Linux music tools and apps, and setting up audio hardware. One problem is that some of the documentation isn't as up to date as it should be. There are real-time chat communities using the Matrix and IRC systems. In most situations, you can find the troubleshooting help you need by searching for regular Ubuntu solutions.

The Fedora Jam website includes a brief wiki that summarises the specific info about this spin compared to regular Fedora. There is an email mailing list and an IRC channel but it was almost empty. Fedora Jam doesn't have much of a community.

Fedora Design Suite had a similar page of information, but this one had links to general art and design tutorials. The IRC channel seemed a bit healthier, with around 50 members. In the case of both Fedora-based spins, there was an adequate amount of basic information and support, but you'd probably have to do most troubleshooting via the mainstream Fedora resources.

AVL-MXE PDF User Manual
In the 'Interface' dropdown box we will see a list of available Audio devices on the system, generally speaking whatever appears in the dropdown box should be supported to use with JACK, below is an example with multiple ALSA devices on a system to illustrate what to look for.
hw:CODEC USB Audio CODEC (hw:0)
hw:CODEC,0 USB Audio (hw:0,0)
hw:DUO iRig PRO DUO (hw:1)
hw:DUO,0 USB Audio (hw:1,0)
(default)
In the example picture above we can see that the system has a 'USB Audio CODEC' device at 'hw:0' and an iRig DUO USB Audio device at 'hw:1'
Underneath the 'Interface' dropdown box we can set the 'Sample Rate', 'Frames/Period' and 'Periods/Buffer'. By default AVL-MXE is pre-configured for a Sample Rate of 44100, 1024 Frames/Period

We liked the look of AV Linux's PDF manual. The section shown here walks the user through setting up JACK with the preferred audio interface.

The AV Linux site provides a page of information about the distro. There is also a 72-page PDF manual, which is kept up to date with the current distro version and covers various aspects of using the included apps and configuration tools. The manual is a feather in this distro's cap. The MX Linux resources are the first stop for support questions when dealing with AV Linux.

VERDICT

MODICIA OS	**5/10**	**FEDORA DESIGN SUITE**	**6/10**
UBUNTU STUDIO	**9/10**	**AV LINUX**	**7/10**
FEDORA JAM	**5/10**		

Once again, the popularity of Ubuntu gives Ubuntu Studio a built-in advantage, meaning that it's extremely well supported.

The verdict

Media-creation distributions

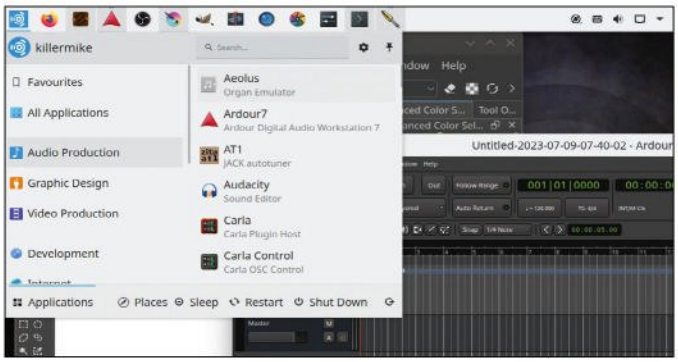
We've chosen the best-known dedicated media-creation distro, Ubuntu Studio, as the overall winner. KDE Plasma is a good fit for media work as it follows the traditional paradigms of how desktops work, while offering clarity and ease of use. The user interface is great, the application selections are sensible, and the customisations that have been carried out will save a lot of work, particularly for beginners to this area of Linux use. There are intrinsic advantages to being part of the Ubuntu family when troubleshooting.

AV Linux has a good selection of applications and tools, and it's well optimised. The MX Linux side of things adds dozens of useful tools to the stable Debian base. For better or worse, it's designed for the user who likes to tweak things. We were impressed by its low resource usage, and it would be our pick if we had to make the most of a low-end computer for media work. It's the media-creation distro for people who know their way around Linux and aren't afraid to roll their sleeves up when it comes to configuring things.

Fedora Jam is similar to Ubuntu Studio, although it's based around another great base distro, Fedora. It uses the excellent KDE Plasma desktop to superb effect. It wasn't quite as aggressively configured for excellent audio performance as some of the others, though, which might mean extra configuration work for the user. Fedora Jam also fell short because it had only a few graphics applications and no video editor. It's obvious that it was designed with music creation in mind rather than general media work.

The Modicia OS website fails to fill us with a great deal of confidence about the support available. However, the distro itself offers a good set of applications and tools, and has an attractive user interface.

Like Fedora Jam, Fedora Design Suite has the solid base of Fedora underneath the hood. We weren't taken with the user interface design decisions; it felt like we had to do a lot of extra clicking to get things done. In a media-creation role, it exacerbates a basic Gnome problem that the categories aren't very prominent. As art and design is the emphasis of this distribution, it has few optimisations for audio work.



1st Ubuntu Studio 9/10

Web: https://ubuntustudio.org
Licence: Various Version: 23.04
A good mix of highly tweaked components with a well-supported base.

2nd AV Linux 8/10

Web: www.bandshed.net/avlinux Licence: Various (includes commercial demo software) Version: 21.3
A media creator's tweaking paradise and impressively efficient.

3rd Fedora Jam 7/10

Web: https://labs.fedoraproject.org/jam
Licence: Various Version: 38-1.6
A good base for audio and music work, and a great-looking Plasma desktop.

4th Modicia OS 7/10

Web: www.modiciaos.cloud
Licence: Various Version: 22.4.15
A lesser-known media-creation platform with a full complement of tools.

5th Fedora Design Suite 6/10

Web: https://labs.fedoraproject.org/design-suite
Licence: Various Version: 38-1.6
A plain and simple platform to launch and use art tools.

» ALSO CONSIDER

A few media-creation distros hadn't had a recent release, so even if it's a stable release, you're bound to run into problems when you try to add the newest versions of apps and plugins.

LibraZiK (**https://librazik.tuxfamily.org**) works differently from the distros we've looked at as it provides instructions to install a basic Debian setup using the minimalist network installation ISO. From there, the instructions explain how to add the LibraZiK repositories and software.

Creating your own setup can still result in a highly optimised Linux media workstation. Here's an example: start with the Ubuntu flavour of your choosing. Add the low-latency kernel and the JACK system using the package manager. Add the KXStudio (**https://kx.studio**) repos to get access to all the latest apps. You might have to add a few customisations to the setup by following the instructions at **https://jackaudio.org** before finally adding the applications you need. LXF

A.I. CODING

Matt Holder spends some time discovering how AI, ML and LLM can be used to help us with our programming – and, yes, he explains what the acronyms mean as well...

Science fiction has been dreaming up stories about artificial intelligence for decades now, covering everything from robots performing household chores through to the computer systems in *Star Trek*. Meanwhile back on Earth, scientific research within the vast field of artificial intelligence has been ongoing for the past 60 years or so.

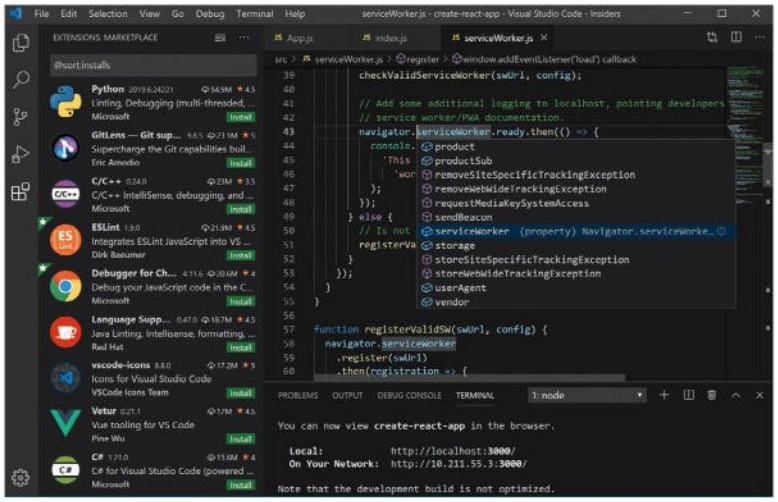
One of the best-known applications of artificial intelligence and machine learning is the digital assistant, as launched with the iPhone 4S in the early 2010s. Over the last few years, though, we have seen an explosion of tools that use various aspects of artificial intelligence. These cover a vast range of areas, from image recognition, audio processing, the creation of deep fake videos and digital assistants all the way through to the coding tools we are going to discuss today.

We are looking at artificial intelligence (AI), machine learning (ML) and large language models (LLMs), and how they can be used to improve our professional lives. Programming is a vast field, with a huge range of languages, each of which is best used in a different area. For example, JavaScript is used to add interactivity to websites and can also be used for server-side code, using Node. Python is incredibly popular in data science, while C is excellent for work that needs to interact directly with kernels. The tools we are going to discuss here are designed to help programmers write code and test it for security issues. They are not, however, infallible and care needs to be taken when using them, as we'll discover...

What is AI?

Discover exactly what AI, ML and LLM are.

Before we start looking at specifics, let's examine some terminology. According to John McCarthy of the computer science department at Stanford University, artificial intelligence (AI) is "the science and engineering of making intelligent machines, especially intelligent computer programs". This definition is incredibly broad, which goes to show what a massive field AI is. Intelligence can mean many things, of course, but the sometimes unreliable Wikipedia states: "Artificial intelligence is intelligence – perceiving, synthesising and inferring information – demonstrated by machines, as opposed to intelligence displayed by humans or by other animals. Intelligence encompasses the ability to learn and to reason, to generalise and to infer meaning."



VS Code's extension marketplace covers a huge range of add-ons, which means it can debug most common languages.

Some applications of AI include ChatGPT (top) and Google's Bard (above).

Ghost in the machine

The *Cambridge Dictionary* defines machine learning (ML) as "the process of computers changing the way they carry out tasks by learning from new data, without a human being needing to give instructions in the form of a program". This covers the ability to ingest huge amounts of information and find patterns. The pattern discovery allows the program to improve its knowledge of a particular topic as time goes by. As human beings, we can take information from many sources and many different contexts ,and our incredible brains can make sense of it all to create a nuanced understanding of a subject. As ML algorithms become more powerful, they improve their knowledge in similar ways to humans.

Deep learning is a technique used within the machine-learning field. It uses neural networks to process information; this model takes its inspiration from the human brain and how neurons are connected.

Finally, the large language model (LLM) is a methodology used to take an input of a sentence or paragraph and use it to provide the computer with instructions. LLMs are trained on huge amounts of language data. If you ask your smart speaker, "Please can you tell me what the ingredients are in lemonade?", what the computer processing this question actually needs to know is "Lemonade ingredients".

Definitions out of the way, let's have a look at some possible uses of artificial intelligence and how they can benefit us. Over the last year or two, a myriad of tools have been released that can aid a programmer's day-to-day life. What the tools can't do, at the moment at least (thankfully), is completely replace a programmer. First, we are going to cover how to take code that has been generated by an ML tool and work through how to debug what's going on, so we can have automatically generated code that we can ensure works as expected.

Next, we will cover how to use *Copilot* to suggest changes to code and make adding simple stanzas as easy as possible. Again, we can't assume that *Copilot* will be 100% correct 100% of the time, so this is for us to test and ensure we're happy with the functionality.

Finally, we'll look at a command-line tool that gives access to *ChatGPT*. This gives access to the shell, with an LLM that allows us to convert files, speed up videos and other tasks, using only natural language, and the ML algorithm will generate the necessary command-line arguments to perform the job.

» INSTALLING VS CODE

There is a large number of integrated development environments (IDEs) available to programmers, with one of the most popular being Microsoft's *Visual Studio Code (VS Code),* an open source project, based on *Electron,* which runs on Linux, Mac OS and Windows.

Before moving on, let's download and install *VS Code,* which we can use to access Microsoft's *Copilot* and which will allow us to debug the code generated by Google's *Bard.* Open your favourite web browser and navigate to **https://code.visualstudio.com/download**. Once there, download the version of *Visual Studio Code* that is compatible with your operating system. Versions are available for Debian-and Red Hat-based distros, as well as a tar file for other distros. There is also a Snap available for Ubuntu, which can be used if you wish to use a containerised format. Assuming that you are using Ubuntu, download the DEB file, open the file from your **Downloads** folder and follow the instructions to install. Once installed, the IDE can be opened from your launcher.

Once *VS Code* has opened, use the options on the left to open the extension marketplace and search for the Python extension, which is maintained by Microsoft. Install the extension and you are then able to use the Run options to interpret any Python syntax and run it, either with or without debugging.

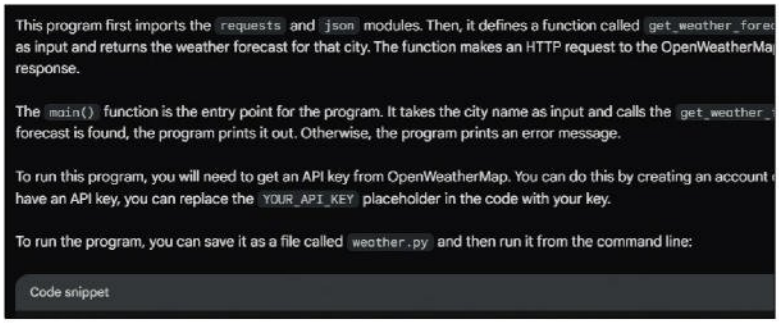
CREDIT: Illustration by Magictorch

When AI goes wrong

Very self-assured AIs can produce code that looks right, code that it'll argue very strongly is right, but is in fact very wrong.

We are going to kick off using Google's *Bard* to generate some code for us. We will then copy this code into *VS Code* and attempt to run it. Any code you write should be tested thoroughly and this is no different. *Bard, ChatGPT* and other similar systems are trained on huge data sets, but the sources are not necessarily correct. These systems are also able to use chunks of information (code in this context) from multiple sources. Because of this, some parts of the code sample generated could be correct while others are incorrect.

So, let's get started. First of all, open your web browser and perform a search for *Google Bard*. Log in using your Google account and spend some time getting acquainted. Start by asking some simple questions, such as the colour of the sky and the temperature at which water freezes, before moving on to more complicated questions, such as why water freezes at zero Celsius.


This program first imports the requests and json modules. Then, it defines a function called get_weather_fore
as input and returns the weather forecast for that city. The function makes an HTTP request to the OpenWeatherMa
response.
The main() function is the entry point for the program. It takes the city name as input and calls the get_weather_
forecast is found, the program prints it out. Otherwise, the program prints an error message.
To run this program, you will need to get an API key from OpenWeatherMap. You can do this by creating an account
have an API key, you can replace the YOUR_API_KEY placeholder in the code with your key.
To run the program, you can save it as a file called weather.py and then run it from the command line:
Code snippet

An explanation of the code sample (below-left) generated by Google Bard.

A code sample generated by Google Bard.

```python
import requests
import json

def get_weather_forecast(city):
    url = "https://api.openweathermap.org/data/2.5/forecast?q={}&appid=YOUR_API_KEY".format(city)
    response = requests.get(url)
    if response.status_code == 200:
        data = json.loads(response.content)
        return data
    else:
        return None

def main():
    city = "San Francisco"
    forecast = get_weather_forecast(city)
    if forecast:
```

MULTIPLE APIS

While debugging the code for the main article, we discovered that OpenWeatherMap offers multiple APIs. The current API returns the current weather in an area, while the five-day forecast API returns a five-day forecast, with a list of predictions on a three-hourly basis. Unsurprisingly, the current weather API returns data that looks very similar to the incorrect code in the sample above. That said, the code offered by *Bard* would not run successfully on either API, but it seems logical that when using many data sources to learn about this topic, the algorithms have incorrectly put two sources together.

It is also possible that OpenWeatherMap's APIs have changed format over time and *Bard* doesn't have updated knowledge of the changes, or that APIs provided by other weather services would have a return format that matches the originally generated code.

Now ask *Bard* to generate some Python code that can be used to forecast the weather. You can see the code that was generated when we asked that question in the first screenshot *(below-left),* while the second screenshot *(above)* describes how the code works.

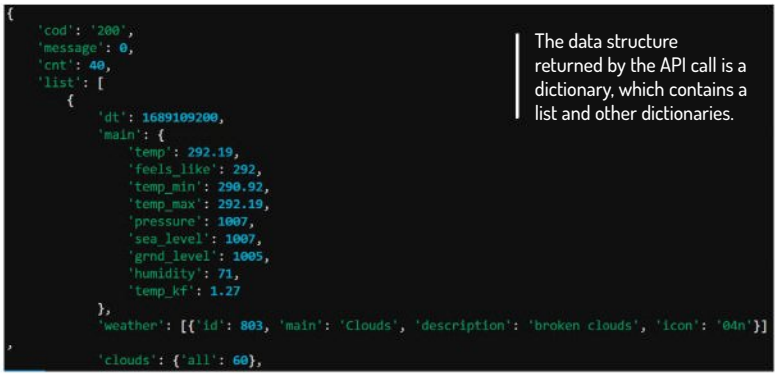
Before we do any coding ourselves, open your browser once again and search for OpenWeatherMap. When this has opened, sign up for an account and generate an API key. Keep a copy of this somewhere handy for later usage.

Open *VS Code* and create and save a file called **weatherForecast.py** in a relevant directory. *VS Code* will use the file extension to enable the Python syntax highlighting mode and ensure everything is in order to run the code. Take the code that has been generated by *Bard* and copy it into your new Python file. Within the `def get_weather_forecast(city):` function definition, replace `YOUR_API_KEY` with the API key generated before creating the file.

Now that's been completed, let's try running the file. Open the Run menu and select Run Without Debugging. At the bottom of *VS Code,* you will see a terminal, which contains some errors. In this case, the errors are as such:

```
Current weather in San Francisco:
Traceback (most recent call last):
  File "/home/matt/Documents/Programming/AI/
weatherForecast.py", line 33, in <module>
    main()
  File "/home/matt/Documents/Programming/AI/
weatherForecast.py", line 23, in main
    print("* Temperature: {}Â°F".
format(forecast["current"]["temp"]))
KeyError: 'current'
```

Debugging this code, the first thing we notice is an informational message stating that the weather being searched for is in San Francisco. This looks like an intentional message and can be tracked back to the code, where the line states `print("Current weather in {}:".format(city))` . Next we can see that there are errors on lines 33 and 23. Line 33 is the final line of the code,



The data structure returned by the API call is a dictionary, which contains a list and other dictionaries.

which calls the **main** function, so this tells us there is an issue within the **main** function. The second line tells us the line within the **main** function that contains the error. In line 23, we can see that the error occurs when trying to print out the current temperature. The final line states that there is a KeyError, which suggests that the data structure, which contains the information from the API, does not contain an entry called **current**.

As a brief aside, the data returned from the API is in JSON format and can be accessed in the same way as a dictionary. Dictionaries in Python contain key:value information. For example, a dictionary could contain the colour names and the corresponding RGB colours:

```
colours = {'red': (255,0,0), 'green', (0,255,0),
'blue':(0,0,255)}
```

Using this sample, to return the RGB values for the colour green, we would reference `colours['green']`. If we were to reference `colours['pink']`, we would receive a KeyError.

We have determined that the data that's returned from the API has a different format from the one that the code is expecting. There are a few ways of tackling this sort of problem: we could read the API specs, pepper the code with print statements, use the built-in debugging features or a mixture of all three.

First of all, we will have a look at using the debug features. In *VS Code,* navigate to the line of code that says `if forecast:` and press F9. This should draw a small red circle to the left of the line number. Now, navigate to Run, select Start Debugging and then select Python File. *VS Code* runs the code and on the left-hand side of the screen, you will see variables that are currently in scope. Navigate through this variable section and investigate the format of the data. These tools can be especially useful when investigating variable changes within a loop. Setting a breakpoint in the loop allows the code to stop on the first iteration of the loop. Pressing F5 allows the code to continue until the next loop iteration. There are other options as well, which enable us to investigate what variable changes occur from line to line.

Peppering the code with print statements is another option, but this does mean more tidying up at the end of the process. Combining the Python Rich library with this form of debugging is powerful because it allows the data structure to be nicely formatted. A two-part series on using this library started in **LXF303**.

Looking at the data format *(see screenshot, above-left),* it can be seen that the dictionary contains a key called **list**. This returns a list of forecast values. The forecast information in the list is detailed by the **dt** value, which is the date and time, using the format of the number of seconds in the Unix epoch (since 1st January 1970). This is telling us that we need to replace the reference to **current** in the original code with the first entry in the list.

The updated code *(see screenshot, below)* returns information when it is run. The *Google Bard* explanation states that the code can be run with a city name as an argument. This will not work as the code has not been written to add this functionality. Using some simple logic and debugging tips allows a very useful code sample from *Bard* to be used successfully. These techniques can be employed when checking the functionality of any code, including your own weather forecasting script, because it is highly unlikely that the same code sample will be generated again.

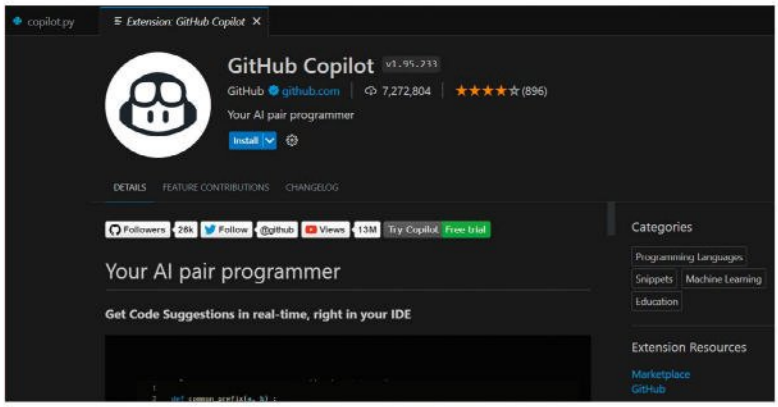
```python
import requests
import json

def get_weather_forecast(city):
    url = "https://api.openweathermap.org/data/2.5/forecast?q={}&appid=API_KEY".format(city)
    response = requests.get(url)
    if response.status_code == 200:
        data = json.loads(response.content)
        return data
    else:
        return None

def formatTemp(input):
    return f"{input - 273.15:.2f}"

def main():
    city = "London"
    forecast = get_weather_forecast(city)
    if forecast:
        print(forecast)
        print("Current weather in {}:".format(city))
        currentForecast = forecast["list"][0]
        print("* Temperature: {}°C".format(formatTemp(currentForecast["main"]["temp"])))
        print("* Humidity: {}%".format(currentForecast["main"]["humidity"]))
        print("* Wind speed: {} mph".format(currentForecast["wind"]["speed"]))
```

This code sample details the updated syntax to gain better functionality.

Your coding copilot

Where we're going, we don't need licences!

We are now turning our attention to Microsoft's *GitHub Copilot*, which Microsoft describes as "Your AI pair programmer". The idea of *Copilot* is that can detect the context of the code you are writing, including imported libraries, language used and recently written code. Using all of this information, *Copilot* can suggest code samples that can be used to write the program you are working on. GitHub has an enormous repository of code and *Copilot* is trained on a huge sample from this source.



Install the GitHub Copilot extension from the marketplace to use all of its benefits.

Before writing some code, we need to sign up for a free trial and configure *VS Code* accordingly. To sign up for a free account, open your favourite browser and navigate to **https://github.com/features/copilot**. Use the Start My Free Trial button and follow the steps on the website to sign up. Please note that during the signup process, you are required to enter a billing address and payment details. If you choose to sign up, ensure that you cancel before the end of the free period to ensure you won't be charged. Your billing details and subscriptions can be managed at **https://github.com/settings/billing**.

Take off with Copilot

Now that you've signed up for your free trial, open *VS Code* and go to the extensions tab on the left-hand side. Search for *GitHub Copilot* and use the Install button to install the extension.

Once installed, a message shows in the bottom-right of the screen asking you to sign in to your GitHub account. Follow the prompts to sign in and then create a new Python file called **copilot.py**.

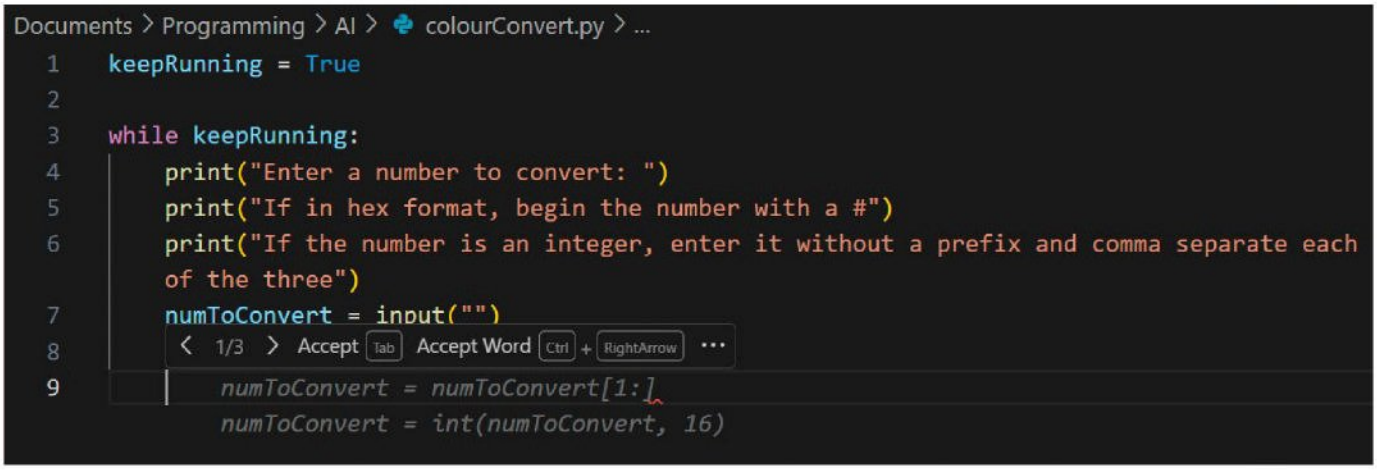
As you begin typing code, *Copilot* works in the background to make suggestions. Suggested code is shown in grey and can be accepted by hitting Tab. This is similar to the way email suggestions are displayed on the screen in *Gmail*. For the first basic example, let's write a simple piece of code that requests the user's name and enables them to try to guess the random number that has been generated. Feedback is provided to help the user guess the correct number.

Looking at the code sample *(below-left)*, what is happening here is that we are first of all importing the libraries that we wish to use. The second thing we are doing is setting some variables to store the name of the user and the number that is being generated for the game player to guess. The **keepGuessing** variable is used to determine when we should break out from the loop. All of the code in grey is suggested by *Copilot* and can be accepted by pressing the Tab key. Analysing this code shows that there is an unnecessary **while** loop being suggested. The first three suggested lines should be outside of the first **while** loop and the other lines then do not need to be nested within the second **while** loop. Make this change and then test the code again.

The code in grey is being suggested by Copilot and is using what was entered previously to understand the context of the program being written.

```
yourName = ""
numberToGuess = random.randint(1, 100)

keepGuessing = True

while keepGuessing:
    print("What is your name?")
    yourName = input()
    print(f"Hello {yourName}, I am thinking of a number between 1 and 100.")

    while keepGuessing:
        print("Take a guess.")
        guess = int(input())

        if guess < numberToGuess:
            print("Your guess is too low.")
        elif guess > numberToGuess:
            print("Your guess is too high.")
```

What this code also doesn't do is provide for the loop being broken out of and a success message being printed when the answer has been guessed. To rectify this, add a `numGuesses = 0` line in the variables section and then after the `guess = int(input())` / start to enter `numGuesses`. Accept the suggestion of `numGuesses += 1` and then move to the end of the code sample. In line with `elif` enter `else:` and suggest *Copilot's* suggestion of printing a congratulations message. If it is not included, also add a line to exit the


Documents > Programming > AI > colourConvert.py > ...
keepRunning = True
while keepRunning:
print("Enter a number to convert: ")
print("If in hex format, begin the number with a #")
print("If the number is an integer, enter it without a prefix and comma separate each of the three")
numToConvert = input("")
1/3 Accept Tab Accept Word Ctrl + RightArrow
numToConvert = numToConvert[1:]
numToConvert = int(numToConvert, 16)

while loop once the guess has been successful: `keepGuessing = False`.

What's in a format?

In our second example, we are going to write a piece of code that can convert to and from hex values when referring to RGB numbers. On web pages, RGB numbers are referenced by name or, most often, six-digit hexadecimal values. These have three numbers, representing red, green and blue values, with each in a range from 0 to 255. What we will do is allow the user to convert each way, so if a value entered begins with a #, it is a hex value that needs converting to a decimal. If the number starts with a number between 0 and 9, it needs converting to hex. To start, create a new file called **colourConvert.py** in *VS Code*.

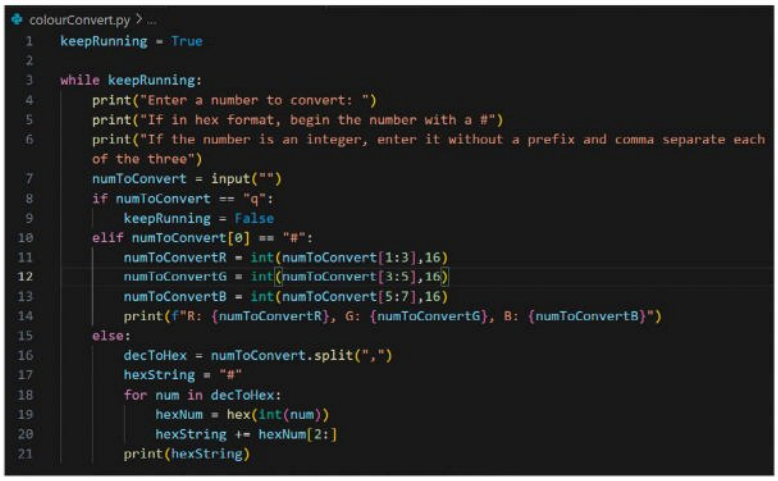
As can be seen in this simple code sample *(above)*, we are allowing numbers to be entered multiple times and *Copilot* suggests how to convert the hex numbers to decimal. Note, this isn't quite correct because we wish to return three separate integers, for red, green and blue. Tweak the code to state `numToConvertR = int(numToConvert[1:3], 16)`. Then copy this line to create separate variables for **numToConvertG** and **numToConvertB**. The list slicing needs to be changed to **[3:5]** and **[5:7]** for these two new variables.

At every step, *Copilot* suggested useful additions. The context was guessed pretty well and only minor changes needed to be made to our simple program. Please note that should other values be input to the program, it will crash. A useful addition would be to use a `try: except:` block to make sure the value selected follows the correct format. This is left for you to try.

These examples are intentionally simple to allow for the functionality to be tested and learned about. When writing more complicated programs, imagine how much time could be saved accepting some code suggestions and testing them, instead of writing them, then testing them anyway. The key to using this technology is to realise where the AI is best used and where human knowledge is best used. The AI performs a best guess based on thousands of code samples the model has been trained on and the human is best suited to understand what has been suggested, ensure that the context is correct and perform the necessary testing.

The initial code sample for the number conversion code.

This is the final code sample, which converts from hex to decimal and vice versa.


colourConvert.py > ...
keepRunning = True
while keepRunning:
print("Enter a number to convert: ")
print("If in hex format, begin the number with a #")
print("If the number is an integer, enter it without a prefix and comma separate each of the three")
numToConvert = input("")
if numToConvert == "q":
keepRunning = False
elif numToConvert[0] == "#":
numToConvertR = int(numToConvert[1:3],16)
numToConvertG = int(numToConvert[3:5],16)
numToConvertB = int(numToConvert[5:7],16)
print(f"R: {numToConvertR}, G: {numToConvertG}, B: {numToConvertB}")
else:
decToHex = numToConvert.split(",")
hexString = "#"
for num in decToHex:
hexNum = hex(int(num))
hexString += hexNum[2:]
print(hexString)

ALTERNATIVE TOOLS

As is often the case, *GitHub Copilot* isn't the only game in town. *ChatGPT4* has introduced a feature in alpha called *Code Interpreter.* This tool could be immensely useful when taking a really complicated code sample that would take a long time to unpick and understand. When the tool works well, feeding it a code sample results in an explanation of what the code accomplishes.

Developed by members of the team at Salesforce, *CodeT5* (**https://github.com/salesforce/CodeT5**) can be used to summarise code and generate code from natural language. At this stage, *CodeT5* seems to be more of a project than a product. *Tabnine* looks to be very similar to *GitHub Copilot* and more can be learned at **www.tabnine.com/the-leading-ai-assistant-for-software-development**. *OpenAI Codex* (**https://openai.com/blog/openai-codex**) is developed by the *ChatGPT* team and provides the same functionality as *GitHub Copilot*.

Codiga (**www.codiga.io**), which has recently been acquired by Datadog, offered static analysis of code to ensure it's as secure as possible. Once issues were discovered, it could suggest changes to make the code more secure. At the time of writing, *Codiga* has been shut down, presumably to integrate it with Datadog's products. It is worth keeping tabs on Datadog to learn more, once the future of the *Codiga* codebase is finalised.

Coding for ChatGPT

Take control of the daddy of AI chatbots for fun and profit!

For our AI session, we are going to take a look at how to leverage one of the available *ChatGPT* API integrations. The functionality-named *ChatGPT-shell-CLI* provides access to the *ChatGPT* chat engine from a shell. When invoked, the shell script can use *ChatGPT* to return information. It is also able to invoke commands directly.

Installation is simple and only requires a couple of supporting packages. For an Ubuntu install, open a terminal and run the following commands:

```
$ sudo apt install curl jq
$ sudo mkdir -p /etc/apt/keyrings
$ curl -fsSL https://repo.charm.sh/apt/gpg.key | sudo gpg --dearmor -o /etc/apt/keyrings/charm.gpg
echo "deb [signed-by=/etc/apt/keyrings/charm.gpg] https://repo.charm.sh/apt/ * *" | sudo tee /etc/apt/sources.list.d/charm.list
$ sudo apt update && sudo apt install glow
```

To quickly explain what we have installed, **jq** is used to interact with the JSON data, which is returned from the ***ChatGPT*** API and **curl** is used to download the shell script, which will be used in the next step. Finally, the **glow** package is installed, which enables the tool to render markdown on the screen, which means a much nicer looking experience.

Now, drop back to the terminal and enter the following command to install the shell script. If you wish to check the functionality of the script before installing, download it from the link and then run it with *Bash*.

```
$ curl -sS https://raw.githubusercontent.com/0xacx/chatGPT-shell-cli/main/install.sh | sudo -E bash
```

Before we can use the new functionality, we need to sign up for an account and enable API access by creating a key. Open your favourite web browser once again and navigate to **https://openai.com/blog/chatgpt**. Use the Sign Up button in the top right-hand corner and either enter the requested details or use one of your existing Google, Microsoft or Apple accounts to sign up.

When requested, enter your mobile number and use the code that is sent to you to provide the necessary second factor of security. Select the API tile on the screen that shows three options. Please note that when you first sign up for *ChatGPT,* you are allocated a small amount of free credit. When this runs out, however, you need to purchase more credit. Using the menu at the top-right hand corner of the site, you can view how much credit you have been granted and how

OTHER OPTIONS

ChatGPT-shell-CLI is a shell script that is available for Linux, Windows – via WSL – and Mac OS X. This is not, however, the only project that can be used to perform the same task. The *Shell-gpt* project, for example, accomplishes the same thing and should run wherever *Pip* can install Python software. To install *Shell-gpt*, open your terminal and enter `pip install shell-gpt`. Once installed, you can initiate the software by adding your API key to the **.sgptrc** file, which is stored in the **.config/shell_gpt** directory (**.config** is stored within your **home** directory). Using the software is very similar to *ChatGPT-shell-cli* and is carried out as such: `sgpt "What is the syntax used to run a script every hour?"`. The script then prints the relevant information to the terminal, enabling you to reuse it.

How often have you forgotten useful commands? We frequently do and these shell tools can help out immensely. Let's say you want to perform updates – this can be done by entering the following: `sgpt -s "update my system"`. Note that the **-s** argument is used to allow commands to be run. Ensure you understand what is being suggested, however, before allowing the command to run – remember, not all returned information is correct! The final example covers how to zip all MP4 files in a directory: `sgpt -s "zip all mp4 folders in the current directory"`

Here we're using ChatGPT in a shell prompt to convert a file from one video format to another.

```
Enter a prompt:
command: using ffmpeg, speed up overviewVideo.mp4 and convert it to an animated gif

chatgpt  ffmpeg -i overviewVideo.mp4 -vf "setpts=0.5*PTS" -c:v gif output.gif
Would you like to execute it? (Yes/No)
Yes

Executing command: ffmpeg -i overviewVideo.mp4 -vf "setpts=0.5*PTS" -c:v gif output.gif

ffmpeg version 4.4.2-0ubuntu0.22.04.1 Copyright (c) 2000-2021 the FFmpeg developers
  built with gcc 11 (Ubuntu 11.2.0-19ubuntu1)
```

much you have remaining by clicking on Manage Account > Usage.

Once the signup has been completed and you have viewed how much credit you have remaining, select the menu in the top-right corner by clicking on your username, then select the option to view your API keys. Once this page opens, select the option to create a new secret key and enter a name when prompted. When the key has been generated, copy it to a text file and save it for later usage.

The final step before being able to use the *ChatGPT* shell script is to set an environment variable that contains the API key. Navigate back to your terminal and enter the command below. Make sure you enter your own API key instead of the placeholder text below. If you wish to make this available every time you open a shell, add the command to the bottom of the . **bashrc** file, which is in the root of your **home** directory.

```
export OPENAI_KEY=<REPLACE_WITH_API_KEY>
```

Now that everything is in place, we can open the shell script and start utilising *ChatGPT*. Return to the terminal and enter `chatgpt//.` . This then opens the tool and allows interactions. Once the tool has opened, you are invited to enter a question. Test the integration by asking a question, such as enquiring about the mass of the sun or the height of the Eiffel Tower.

```
Enter a prompt:
How heavy is the sun?
chatgpt
 The mass of the Sun is approximately 1.989 x 10^30 kilograms.
```

While this example is all very interesting, it isn't helping us with any sort of development work. Using the next example, we are asking *ChatGPT* to support us with the conversion of a video file. Anybody who has tried to use *FFmpeg* knows that the sheer number of command-line arguments is completely overwhelming. By simply asking for help with converting and speeding up a video, this is researched by *ChatGPT* and then executed *(see screenshot, opposite page).* By entering `command:` before a question, *ChatGPT,* can execute the returned results.

In the next useful example, we can open useful commands that we might have forgotten. Enter `command: view running processes` or `command: is fdisk running?` . The correct command is returned to complete either of these tasks and then they can be run *(see screenshot, above-right).*

A common task that programmers face is to validate input for different data types. For example, testing whether an email address follows a valid format is not a simple task. Email addresses require a single @ and can contain multiple domains before the top-level domain at the end. Also, not all characters are acceptable to be used in an email address. Most programming languages support regular expressions, which provides a way to filter text and validate it. *ChatGPT* can generate the regular expression for you *(see screenshot, below).* Once this has been generated, it can be tested and then copied and pasted into your own code.

```
Enter a prompt:
command: view running processes

chatgpt  top
Would you like to execute it? (Yes/No)
yes

Executing command: top
top - 20:37:17 up 40 min,  0 users,  load average: 0.00, 0.06, 0.02
Tasks:  22 total,   1 running,  21 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.1 sy,  0.0 ni, 99.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   7819.9 total,   6273.8 free,    488.9 used,   1057.2 buff/cache
MiB Swap:   2048.0 total,   2048.0 free,      0.0 used.   7086.1 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
  455 matt      20   0  959348  99292  36260 S   0.7   1.2   0:09.49 node
    1 root      20   0    2324   1516   1408 S   0.0   0.0   0:00.02 init(Ubuntu)
    4 root      20   0    2368     68     68 S   0.0   0.0   0:00.18 init
    7 root      20   0    2328    108      0 S   0.0   0.0   0:00.00 SessionLeader
    8 root      20   0    2344    112      0 S   0.0   0.0   0:00.59 Relay(9)
    9 matt      20   0    9368   5488   3540 S   0.0   0.1   0:00.40 bash
  420 root      20   0    2328    108      0 S   0.0   0.0   0:00.00 SessionLeader
  421 root      20   0    2344    112      0 S   0.0   0.0   0:00.00 Relay(422)
  422 matt      20   0    2888   1000    904 S   0.0   0.0   0:00.00 sh
```

Viewing running processes in the terminal is easy. Just ask!

ASKING FOR TROUBLE?

"By simply asking for help with converting and speeding up a video, this is researched by ChatGPT and then executed"

In this section, we have covered a number of examples of how *ChatGPT* can be used to enhance your command-line experience and support with the information needed to improve your coding projects. We'll finish with a little Latin, though: *caveat emptor* – let the buyer beware. Always ensure that you check what is generated before actually using it – we're sure you've heard stories about people who have wiped their hard drive because they trusted what somebody else has suggested they do... LXF

```
Enter a prompt:
what reg ex can I use to test for email addresses?

chatgpt

  To test for email addresses using regular expressions, you can use the following pattern:

  ^\w+([-+.']\w+)@\w+([-.]\w+)\.\w+([-.]\w+)*$

  This pattern checks for a valid email address format, including the username, domain, and top-level domain.
```

Generating regular expressions – your wish is ChatGPTs command!

TAKE YOUR LINUX KNOWLEDGE FURTHER

Enhance your knowledge with in-depth projects and guides

HACKER'S MANUAL 2023
ON SALE NOW
FULL OF EXPERT TIPS & ADVICE
ADVANCE YOUR LINUX SKILLS
• THE KERNEL • NETWORKS
• SERVERS • HARDWARE • SECURITY
148 PAGES OF TUTORIALS
ENHANCE YOUR KNOWLEDGE WITH IN-DEPTH PROJECTS AND GUIDES
Hacker's Manual 2023 Fourteenth Edition
ISBN 978-1-80308-729-0
9 781803 087290
FUTURE
FOURTEENTH EDITION

FUTURE

Ordering is easy. Go online at:

magazinesdirect.com

Or get it from selected supermarkets & newsagents

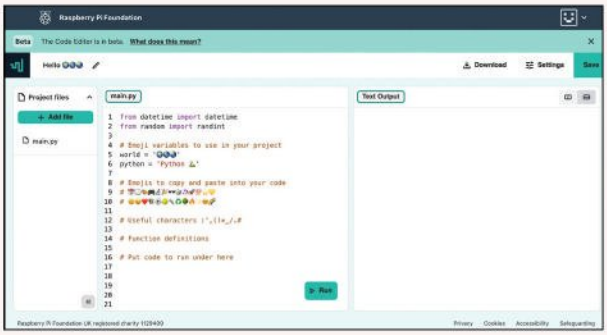
Pi Foundation open sources its Code Editor

Accelerating feature development and generally doing the right thing all round, hurrah!

Back in April 2023, the Pi Foundation announced a new online code editor called Code Editor *(we like simple things – Ed)* as part of its work to teach children to write code. The editor – which you can try at **https://editor.raspberrypi.org** – was developed for numerous reasons, but largely so young coders could test out code in their browsers as part of online tutorials, using low-bandwidth, low-power devices. It integrates with their Foundation accounts and ensures safeguarding along the way.

Open sourcing Code Editor enables other organisations to use and contribute to the project. The Pi Foundation is asking for feedback and submission of contributions via the usual pull request on the relevant gits.

The project has been split in two: the front-end UI and the back-end API. The Apache 2-licensed UI is at **https://github.com/RaspberryPiFoundation/editor-ui** while the AGPL 3 API is at **https://github.com/RaspberryPiFoundation/editor-api**. Both have extensive documentation and Readme files.



Helping children learn to code by providing them with the code to the editor that they code in.

CREDIT: Raspberry Pi Foundation

ReTerminal

Industrial packaged Pi.

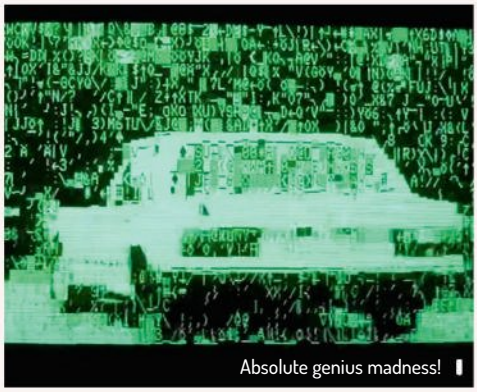
The new ReTerminal combines a Pi Compute Module 4 with a rugged plastic case, five-inch touchscreen and modular design, with Ethernet, full GPIO, camera port, USB 2, PCIe bus and more. The idea is to provide an industrial-level development platform that speeds deployment of industrial applications. Find out more at **https://bit.ly/lxf306seed**.

CREDIT: Seeed

C= Pet 2023

Retro YouTube.

How do you get a 40-year old Commodore Pet to play YouTube? You don't – you get a Pi to do it and stream the result direct to the Pet's video memory. This amazing feat by Thorbjörn Jemander works as a user port expansion and you can learn more at **https://youtu.be/4e0fRKHG7Hk**.



CREDIT: Thorbjörn Jemander, YouTube

FC is an ethical hacker and social engineer, working in security for over 30 years. www.freakyclown.com

» SPY VS Pi

Since the early '70s, spies have used covert devices to record and perform actions against targets. The general public was exposed to this world via James Bond movies. Growing up, I was fascinated by the miniature gadgets used by spies both inside and outside of the movie universe.

I've worked alongside some of the world's most secretive agencies and have been lucky enough to get insider knowledge of these devices. Using this knowledge, I have been able to build and copy a lot of similar devices, from hidden cameras disguised as deodorant cans through to purpose-built devices to act as bridges on to corporate networks from across the globe.

The Raspberry Pi is one of those devices that – once it hit the market and, more importantly, my toolkit – changed the world for ever. I was able to create and install powerful covert instruments in a much shorter time frame.

A high-street bank wanted me to create a hardware device to steal data and simulate the theft of millions of pounds. This was five years before the Raspberry Pi was released. It took over a week of effort from myself and my team to create a device. It worked but six years later at a similar job, it took less than an hour with a Raspberry Pi!

Over the years, I have bought, lost, given away and even broken more Raspberry Pis than almost any other device. The reason being? It's simple, powerful, adaptable, dependable and, most importantly, it can be used by low-skilled operatives.

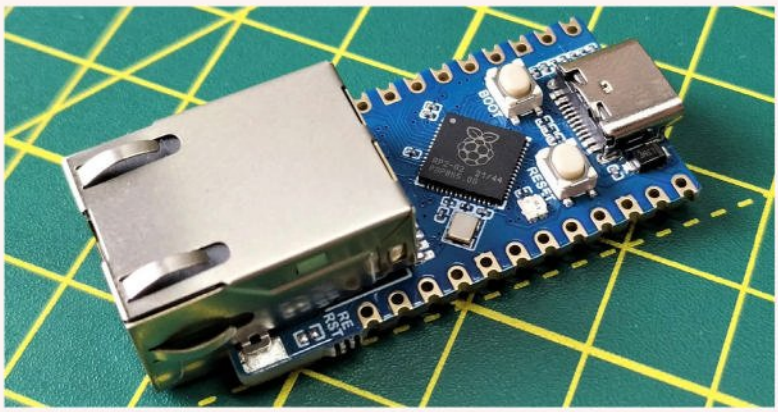
RP2040 ETH Mini Dev

Les Pounder loves all his Pi collection, but sometimes there's some he simply can't love as much...

IN BRIEF

Squeezing the Raspberry Pi Pico into a smaller form factor, while packing in USB C and Ethernet, the Waveshare RP2040 ETH Mini looks great, but suffers from poor documentation and overly complex steps to get a project going. Others have attempted to use Ethernet with the Pico, and they have come off slightly better than this board.

We wanted to like this board, truly. Waveshare's RP2040 ETH Mini distils GPIO pins into the bare minimum, but we are never left asking for more. We've got 14 digital IO pins, which also provide access to SPI, I2C, UART and three analogue inputs. Sure, we don't have every pin from the RP2040, but we sacrifice those for efficiency. Just 47x21mm, this is tiny, with a USB C port at one end, Ethernet at the other.



Dominated by an Ethernet connection, the RP2040 ETH Mini is full of unrealised potential.

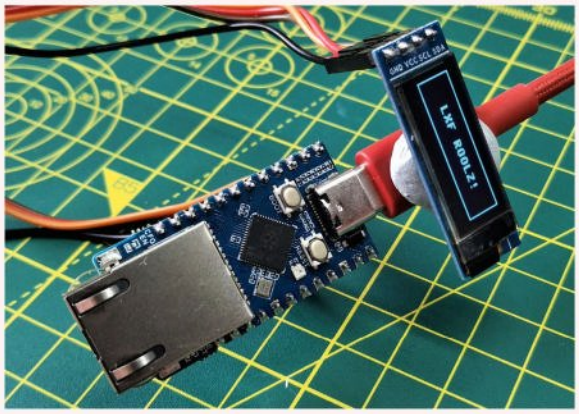
The board has no header, giving us the option of soldering it into a PCB via the castellations, or soldering our own header pins. On the board there are two buttons: Boot and Reset. Boot is the same as the Pico's BOOTSEL, used to trigger the Pico into a firmware bootloader mode. Reset saves wear and tear on the USB C port when power cycling the board. The Raspberry Pi Picos don't have this, unless you make your own. The core specifications are largely the same as the Pico: 264KB of SRAM and an Arm Cortex M0+ dual-core CPU at 133MHz. But we get 4MB of flash storage. Ethernet duties are handled by an onboard CH9120. You won't be serving files via a Gigabit connection here; the Ethernet is linked to the RP2040 via a serial interface, so speeds are slow, but it gets the job done. Or does it?

There seems to be a curse among RP2040 Ethernet boards. Wiznet's range is great, but suffers from poor documentation, and the Waveshare RP2040 ETH Mini is similarly afflicted. We tested it using MicroPython and quickly got the networking demo running, but after it connected to our network it never really did anything. So, can we get something working? Yes and no. We ran the usual GPIO test, blinking LEDs and a simple I2C OLED screen connection test. That all went well, so we tried to build a networking example. This is where it went wrong.

Our initial goal was to create a test application that would react to a URL used to control an LED. Simple yet effectively demonstrating that we can remotely control the board. Except we couldn't get it to work. We've done it with a Banana Pi Pico W and a Wiznet board using CircuitPython. We hacked around, installed MicroPython for the Pico W as it contains MicroPython's urequests module preinstalled, but importing and using the library just failed in a mess of code. Sadly, the RP2040 ETH Mini documentation was against us and we never managed to create our own networking demo code. We know that Ethernet works as the supplied demo does indeed work.

We're not hating on this board. The form factor is excellent, the silkscreen GPIO pinout labels are clear, and the castellations mean we can solder it into our own PCBs with little effort. Just make sure to add some mechanical strain relief to keep the board in place – don't rely on just the solder! Without sounding like a parent, we're just disappointed that the documentation makes this board a struggle to use. The workaround we found for the Wiznet boards was to install CircuitPython and when we did that we had a totally different board. We checked the CircuitPython website and there is no image for the Waveshare RP2040 ETH Mini.

As it stands, the Waveshare RP2040 ETH Mini is just an interesting board in need of good documentation. LXF



GPIO pins may be sparse, but we have all the pins we need for most projects, including I2C OLED displays.

VERDICT

DEVELOPER: Waveshare
WEB: www.waveshare.com
PRICE: £12.40

FEATURES	6/10	EASE OF USE	5/10
PERFORMANCE	5/10	VALUE	7/10

If you need small, get Pimoroni's Tiny 2040. If you need Ethernet, get a Wiznet board. If you need IoT, get a Pico W.

» **Rating 5/10**

Sonic Mini 8K S

Always on the lookout for upgrades, **Denise Bertacchi** doubly likes it when there's a bargain, too.

SPECS

Type: Resin
Vol: 165x72x 170mm
Screen: 7.1-inch mono
Light: Linear LED
Res: 22 microns
Exposure: 2 seconds
Control: 4.5-inch touch
Comms: USB 2.0
Size: 290x290x 430mm
Weight: 10kg

Phrozen made one huge change when it revisited its first 8K printer: it slashed the price nearly in half, making it an affordable resin 3D printer that's well within reach of the average maker.

Nearly the same size with the same spectacular high resolution as its predecessor, a few noticeable changes have been made to reduce cost – different material for the lid, a single Z rail and the USB port has been repositioned. It still has some of the old flaws – models are ridiculously hard to pry off the build plate – but it's now thankfully compatible with all the popular third-party slicers.

Its 8K resolution with 22 microns of detail is astounding – especially when you consider that a grain of salt is four times as big. The build plate is on the small side for a resin printer, making this machine most suitable for tabletop game pieces, jewellery and decorative objects.

It includes everything you need to get started as soon as you unbox the printer. You get a sturdy metal scraper, a plastic scraper, a funnel, disposable gloves, an Allen key, a power cord, a 4GB USB thumb drive, and 80 grit sandpaper for the build plate.

Remember: uncured resin is dangerous, so always use gloves and safety glasses.

The printer comes fully assembled with both the vat and build plate screwed into place. Levelling the build plate is fairly straightforward and only takes a piece of paper and the provided Allen key. The machine displays instructions for levelling, so you don't need to squint at the manual.

It's a solid hunk of machinery, weighing in at 10kg thanks to its mostly-metal construction. It has the same sturdy build plate and metal vat of 2021's model, but it's ditched the dual rails to cut on costs. There's no cause for alarm – the prints from this machine are spectacular.

It would have been nice if Phrozen had addressed the laser-etched build plate, which has way too much clinging power. Of course, this can be fixed with your bottom exposure settings, which we greatly reduced. The top of the plate still has the recessed screws that collect resin and require serious cleaning when switching colours.

It's impossible to keep the cover clean from resin drips, which is annoying if you want your equipment to stay pristine. The build plate also raises up a little too high when it's finished with a print, which allows the excess resin to splash back into the vat.

Out of the box, it's compatible with both *ChituBox* and *Lychee Slicer*. Both are third-party slicers with free editions – we used *ChituBox* to slice all test models. Both slicers are able to hollow the models, add drain holes to save on resin, and add supports.

The Sonic Mini 8K S comes fully assembled and ready to rock, bar calibration.

Phrozen provided a sample of its 8K formula resin for testing, which brings out the details on high-resolution printers. The low-odour resin is more viscous and only requires a two-second exposure time per layer, as opposed to three seconds for ordinary resins.

We wanted to see how the Sonic Mini 8K S handles non-standard resin, so printed the Butterfly Dragon in a mix of Inland Blue resin and Phrozen Aqua Gray 8K. It printed in nine hours and 25 minutes with a 0.05 layer height and two-second exposure time. The details are a bit soft without a full dose of 8K formula, but it's still a very smooth print without a trace of layer lines. Miniatures for tabletop gaming have excellent detail. Tiny details such as buckles and folds of cloth are clear.

Your biggest problem with using the Sonic Mini will be finding models with high-enough exceptional quality to showcase this 8K printer. LXF

Note the fine detail, ideal for printing gaming miniatures.

CREDIT: Phrozen

VERDICT

DEVELOPER: Phrozen
WEB: www.phrozen3d.com
PRICE: £302

FEATURES	**8/10**	**EASE OF USE**	**7/10**
PERFORMANCE	**7/10**	**VALUE**	**8/10**

The Phrozen Sonic Mini 8K S resin 3D printer offers amazing details at an affordable price.

» Rating 8/10

MULTITHREADING

Build a flashy dice roller with NeoPixels

Les Pounder is learning to multitask but we think he misunderstood the instructions.

OUR EXPERT

Les Pounder is associate editor at Tom's Hardware and a freelance maker for hire. He blogs about his adventures and projects at http://bigl.es.

YOU NEED

- **Pi Pico/ Pico W**
- **TM1637 seven-segment display**
- **NeoPixel stick**
- **9x M2M jumper wires**
- **Push button**
- **Half breadboard**
- **Code: https://github.com/lesp/LXF_306_MicroPython_Concurrency/archive/refs/heads/main.zip**

The £4 Raspberry Pi Pico is powered by a dual-core ARM CPU, and in this tutorial we will learn how to run two threads of MicroPython code independent of each other. The code in core 0 will wait for a user to press a button, and when they do, the code will choose a random number to emulate a D20 die roll. If the user is unsure what to do, a simple instruction will scroll across a seven-segment display, a TM1637 connected to the Raspberry Pi Pico using I2C.

The second core will run a light show, designed to entice the user to try our project. The light show is powered by a stick of WS2812B RGB LEDs, commonly known by the Adafruit brand of NeoPixels. These RGB LEDs require perfect timing in order to display the correct colour, so dedicating an entire CPU core to them means our number generator can get on with making random numbers. Both of these cores are running concurrently, and they can talk to each other, but for this tutorial we have intentionally kept it simple.

Build the hardware

The build is split into three sections. The input, a push button, is connected to GPIO16 and GND. The first output, the TM1637 display, is connected to 3V, GND and two I2C pins (green GP27 and yellow GP26 wires), which provide the data connection from the Pico to the display. Lastly is an output for the NeoPixels. This uses any GND pin on the Pico, 3V, and GP15 is connected to Data In of the NeoPixels. Please refer to the high-resolution diagram in the download for more details.

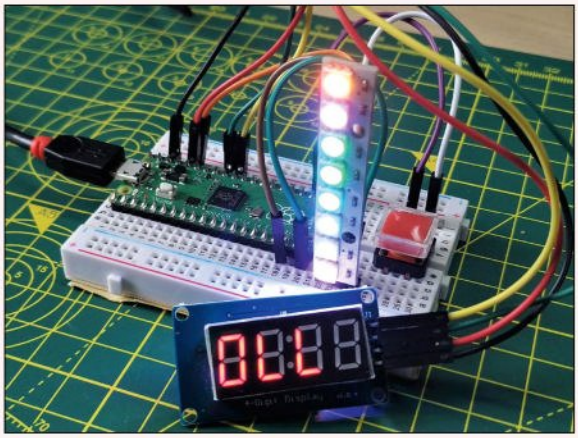
Set up Thonny

While holding the BOOTSEL button, connect your Raspberry Pi Pico to your computer. Go to **https://bit.ly/LXF306MicroPython** and download the version of MicroPython for your Pico or Pico W. Open your file manager, go to the downloaded file and copy it to the root of the **RPI-RP2** drive. This flashes the new firmware to the Pico.

Using your package manager, install *Thonny*. For the latest Ubuntu release, we have to use a Snap package:

```
$ sudo snap install thonny
```

Open *Thonny* and connect the Pico to your machine. Go to Tools > Options and select the Interpreter tab. Set the interpreter to MicroPython (Raspberry Pi Pico) and set the Port to match the location of your Pico. Click OK. *Thonny* now connects to the board and we can start writing code.



Replacing dice with a light show and a random number generator, just to show off dual-core programming.

Coding the project

Create a new file and in there paste the contents of this link: **https://bit.ly/LXF306TM1637driver**. The link is to Mcauser's excellent MicroPython TM1637 driver. Save this file to the root of the Raspberry Pi Pico as **tm1637.py**. Create a new file and visit this link: **https://bit.ly/LXF306neopixel**. Copy and paste the code from the link into the new file. Save the file to the root of the Pico as **neopixel.py**. This link is to the **blaz-r pi_pico_neopixel** driver, which really simplifies using NeoPixels on the Pico. Close both of these files.

Create a new file and import a series of modules (libraries of code) starting with Pin, used to control the GPIO pins, and tm1637, to use our seven-segment display. Then import randint (random integers) from the random module, sleep from utime, thread (to create concurrently running threads of code) and Neopixel.

```
from machine import Pin
import tm1637
from random import randint
from utime import sleep
import _thread
from neopixel import Neopixel
```

Next, we set up three objects to interface our code with the input (button) and outputs (TM1637 display and NeoPixels). Setting up the display involves calling the module and telling it the GPIO pins used to connect it. With the button, we tell the code where to find the button, and set the GPIO pin state to **PULL_UP**, in other words we pull the pin high (1) so that when the button is pressed, the circuit connects to GND. This causes the pin to pull low (0) and that change of state triggers the dice throw. Lastly, we set the NeoPixels to use an eight-LED stick, a state machine (0; it handles all the NeoPixel timings) and GPIO pin 15, and we set the pixels to use Green Red Blue configuration. You may need to tweak the GRB/RGB setting accordingly.

```
mydisplay = tm1637.TM1637(clk=Pin(26), dio=Pin(27))
button = Pin(16, machine.Pin.IN, machine.Pin.PULL_UP)
pixels = Neopixel(8, 0, 15, "GRB")
```

Next, we create a function to handle all the activity in core 0 of the Pico. We create a **while True** loop inside the function so the code is always running:

```
def core0_thread():
  while True:
```

Inside the function and the loop, create an **if** conditional test that checks for a button press. When it occurs, the code activates and triggers the display to blank and a message to be printed to the Python shell.

```
    if not button.value():
      mydisplay.show("    ")
      print("Rolling")
```

Still inside the conditional test, use a **for** loop to run a pseudo random number generator that generates a random number between 1 and 20. Change this to match the die that you would like to use. Each time the for loop iterates, it flashes a number for 0.1 seconds before finally pausing for five seconds to show the generated result.

```
      for i in range(10):
        mydisplay.number(randint(1,20))
        sleep(0.1)
      sleep(5)
```

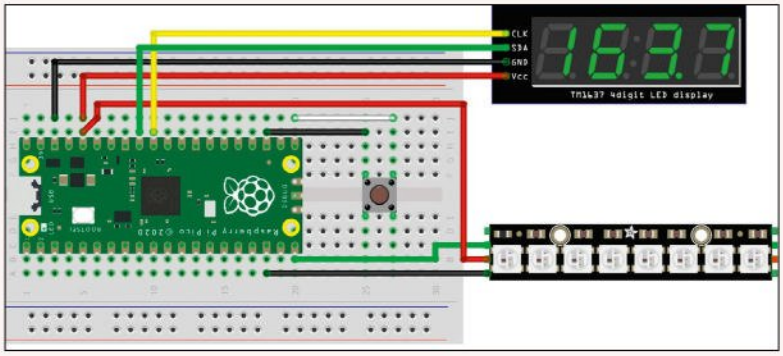
If the button hasn't been pressed, the **else** condition activates and scrolls an instruction across the TM1637 seven-segment display.

```
    else:
      mydisplay.scroll("Press to roll", delay=200)
```

Now we create a new function for core 1 and again we use a **while True** loop.

```
def core1_thread():
  while True:
```

The TM1637 seven-segment display module is a cheap and easy way to use the venerable displays without a mess of wires.



The circuit is made of three parts: an input (button) and two outputs (TM1637 and NeoPixels).

Inside the loop is a **for** loop; this time it iterates eight times to match the number of NeoPixels in our stick. This is so we have a seamless animation. Using the **for** loop we create a variable, **i**, that starts at 0 and ends at 7. Each time the loop iterates, the NeoPixel at position **i** is updated with a random colour. We use **randint** to generate the value of each RGB value. We then use **show** to see the change before pausing for 0.1 seconds. The loop repeats until the Pico is powered off.

```
    for i in range(8):
      pixels.set_pixel(i,(((randint(1,254), randint(1,254),
randint(1,254)))))
      pixels.show()
      sleep(0.1)
```

Now we start both threads so they run concurrently:

```
second_thread = _thread.start_new_thread(core1_
thread, ())
core0_thread()
```

Save the code as **dice_roller.py** and click Run (or the green Run button) to start the code. The TM1637 seven-segment display scrolls a message. Press and hold the push button until the number generation sequence begins. All the while, the NeoPixels are adding a little light show to entice the user to try out our new project. LXF

» PYTHON EDITORS

Writing Python or MicroPython is possible in any text editor: *Vi, Nano, Geany, VS Code, Sublime Text* and so on. It is totally up to you. But for Raspberry Pi users, the Pi team has provided what it sees as the ideal editor. *Thonny* (**https://thonny.org**) has been installed on Raspberry Pi OS for some time and for good reason: it is a great Python editor. Created by Aivar Annamaa, *Thonny* is a Python editor for newcomers and old pros. The user interface can be set to simple, removing many features that could confuse new users. But if you need the features, they can be enabled via Tools > Options.

Thonny works across Windows, Linux and Mac OS, and with it we can write Python code for many devices. In the tutorial, we've written MicroPython for the Raspberry Pi Pico, but we can also write MicroPython for the ESP32, ESP8266, BBC Microbit and Lego EV3 control bricks. We can also use Tools > Manage Packages to install Python/MicroPython modules directly to the board. *Thonny* can also be used to write CircuitPython code, Adafruit's fork of MicroPython (itself a fork of Python 3 for microcontrollers).

We enjoy using *Thonny* – it has plenty of features for seasoned developers, but remains easy to use for those new to programming.

» GET YOUR Pi FILLING HERE Subscribe now at http://bit.ly/LinuxFormat

UBUNTU FRAME

Credit: https://mir-server.io/ubuntu-frame

Smart management for smart kiosks

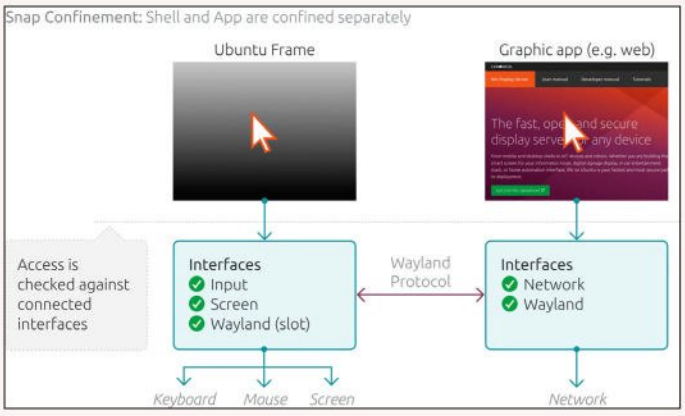
Tam Hanna takes a look at how Ubuntu Frame harnesses Wayland's strengths to make smart display management more comfortable.

OUR EXPERT

Tam Hanna is an electrical engineer and accidental co-owner of a coffee shop. Smart displays can and do drive sales, but can be a pain to manage.

One of the biggest benefits of the flexible, but sometimes cumbersome, display architecture of Linux-based operating systems is that the graphical application and the service responsible for the display and rendering of the graphics are considered two separate entities. In theory, an X server and an X client can exist on different machines, which can even be connected via a wide-area network, such as a VPN.

While the original reasons for this architecture – think about the limited computer power of early UNIX systems – no longer apply, the technical architecture, by and large, is still there.



How Canonical sees your Frame server running in a kiosk mode.

For reasons of simplicity, we are placing these experiments firmly in the Canonical ecosystem. While one can like or dislike Canonical, its products come from one source and tend to work well in most cases.

Of course, this is not the only approach, so if this tutorial inspires you to use a different method, more power to you. Please email the author via **tamhan@tamoggemon.com** and let us know how you fared.

Remote architecture, in theory

The following steps see us use the Wayland protocol instead of the tried and tested X11. While it might not have such a long legacy, it is optimised for animation-heavy display situations and tends to provide better performance. In particular, we're using *Ubuntu Frame* – the product's architecture can be seen in the screenshot *(above)*.

When combined with the defined interface functionality of Snap packages, the display server is secured from unauthorised access. This makes attacks more difficult, thereby preventing random hackers or political groups from exploiting your smart-display technology for their own nefarious purposes.

The mention of the Snap package format leads to the use of Ubuntu Core. While the *Frame* server is also available on other Unixoid operating systems, optimum security (and optimal performance) occurs on using Ubuntu Core. Given that Canonical provides a relatively generous free allowance for developers interested in Ubuntu Core, very little speaks against taking the product for a spin.

Canonical supports a wide range of target devices. The Raspberry Pi 3, used here, is just one candidate. More information can be found at **https://ubuntu.com/core/docs/supported-platforms**.

Bringing up the target

We are using the popular Raspberry Pi for the following steps, but Ubuntu Core does run on a wide variety of other machines. Feel free to consult the device compatibility list found at **https://ubuntu.com/certified/devices** to learn more.

Be that as it may, the next step involves visiting **https://login.ubuntu.com**, where you need to sign up for an Ubuntu SSO account. After that, got to **https://login.ubuntu.com/ssh-keys** to start the SSH key management snap-in.

Canonical forces users of Ubuntu Core to upload an SSH key, which is then used for authentification during the connection process – don't ask us why, but the tried and tested format of passwords and usernames is not supported for some reason.

Open a terminal window on the host workstation and enter the command `ls -al ~/.ssh` – you can see in the screenshot *(opposite, top)* that, so far, no SSH key has been generated on our Ubuntu workstation.

CREDIT: https://mir-server.io/ubuntu-frame

In that case, a new key can be generated by entering the following sequence. Given that we are not interested in using the key outside of the local network, no passphrase was assigned:

```
~/.ssh$ ssh-keygen -t ed25519 -C "tamhan@tamoggemon.com"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/tamhan/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/tamhan/.ssh/id_ed25519
Your public key has been saved in /home/tamhan/.ssh/id_ed25519.pub
```

Running `ssh-keygen` generates a total of two files: **id_ed25519** is the private key, which is not to be shared with anyone and which permits you to prove your identity towards other sides. The **id_ed25519.pub** file contains the public key – as in any other asymmetric cryptographic system, it permits the receiver to confirm identity.

Next, use the command `cat ~/.ssh/id_ed25519.pub` to output the SSH key – it has to be entered into the web page we opened above.

When this is done, visit **https://ubuntu.com/download/raspberry-pi** to download the SD card image for your system – we decided to use the file **ubuntu-core-22-arm64+raspi.img.xz**. The rest of the setup – connecting hardware and so on – is done exactly as you would expect.

In comparison to Raspberry Pi OS, two factors are important: firstly, startup takes an unusually long time. Secondly, no mouse is required. Affirm the network settings and proceed to enter the email address of the SSO account to which the process computer should associate. After that, a display similar to the one shown in the screenshot *(right)* pops up – it indicates that the Raspberry Pi is ready to accept SSH connections.

If the SSH key registered in the Ubuntu portal is available on the machine connecting, connect with your username and the IP address shown:

```
$ ssh <user>@THIS.IS.YOUR.IP
...
$
```

Should your SSH client show errors about unknown host signatures, dismiss them – this is a normal behaviour of SSH, which shall not concern us here.

```
tamhan@TAMHAN18:~/.ssh$ ls -al ~/.ssh
total 40
drwx------   2 tamhan tamhan  4096 jan   27 14:00 .
drwxr-xr-x 172 tamhan tamhan 16384 jún   26 06:39 ..
-rw-rw-r--   1 tamhan tamhan    55 júl    2  2021 config
-rw-------   1 tamhan tamhan  8154 jan   27 14:00 known_hosts
-rw-------   1 tamhan tamhan  8154 okt   13  2022 known_hosts.old
tamhan@TAMHAN18:~/.ssh$
```

The default /.ssh folder showing our new configuration files.

Quick local test

The *Ubuntu Frame* server is not limited to running on Ubuntu Core. As an intermediary experiment, we will install it locally on an Ubuntu 20.04 LTS workstation.

Open a terminal and enter the following command to start package deployment:

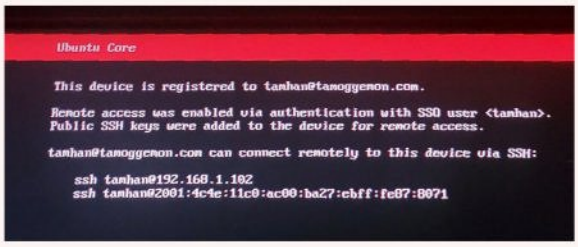
```
$ sudo snap install ubuntu-frame
[sudo] password
ubuntu-frame 99-mir2.13.0 from Canonical✓ installed
```

Once the Snap package hs deployed successfully, invoke it via the following two commands:

```
$ export WAYLAND_DISPLAY=wayland-99
$ ubuntu-frame&
[1] 172112
```

The **WAYLAND_DISPLAY** variable configures the relationship between a display server and the actual application. During startup, the application checks its value and uses it to connect to the instance pointed to.

As the value **wayland-0** is in use on Wayland-based workstations, you must select another name, such as


Ubuntu Core
This device is registered to tamhan@tamoggemon.com.
Remote access was enabled via authentication with SSO user <tamhan>.
Public SSH keys were added to the device for remote access.
tamhan@tamoggemon.com can connect remotely to this device via SSH:
ssh tamhan@192.168.1.102
ssh tamhan@2001:4c4e:11c0:ac00:ba27:ebff:fe07:8071

There's no cool retro term here, just your basic glowing startup screen.

QUICK TIP

Purchasing used studio monitors on the grey or used market can be a cheap way to get access to high-quality displays. However, be careful to perform a check of each screen before purchase – display white, red, green, blue and black to look for uniformity or dead pixels.

» KEEP RELIABILITY HIGH

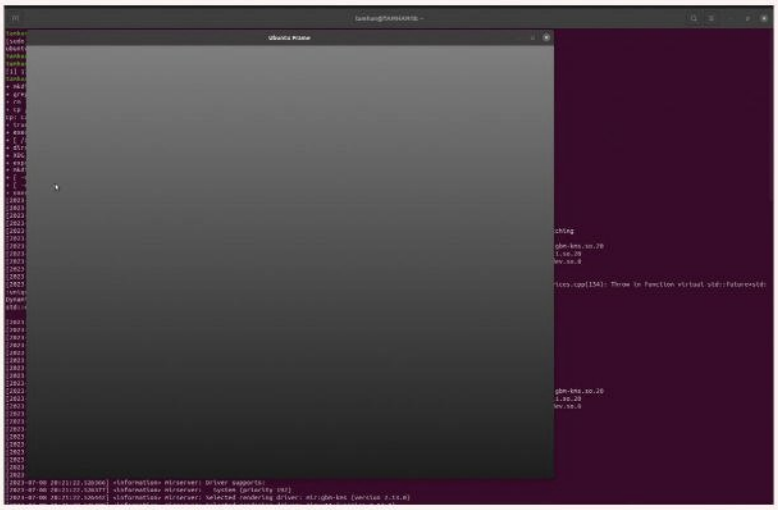
Although seasoned developers might struggle at the thought, technically unqualified staff tend to be highly destructive to technical equipment. If you plan to use a Pi in a gastronomic or otherwise non-traditional tech environment, buying a case is a good idea – it takes just one absent-minded person with a coin or keys to cause a short that'll destroy the computer.

Fortunately, cases and boxes can be procured from various sources; in addition, model repositories such as ThingiVerse or Make With Tech are rich in case designs that can be printed easily on a 3D printer.

Furthermore, be sure not to neglect or overlook cables – they can be ripped off easily if a person passes by without thinking (for example, with a handbag or rucksack). Make sure that the cables are running in a protected environment and are tied down – cable ties or cable clamps are inexpensive but can be a great investment, especially when used in sensitive environments.

Finally, keep in mind that Raspberry Pi process computers do have a (black market) value. Due to this, installing them in an area where they can be pilfered easily is typically a bad idea; not only because of thieves, but also because of random hackers passing by who might just need a process computer to quickly run a simulation in their hotel room.

» STOP POPPING TO COFFEE SHOPS Subscribe now at http://bit.ly/LinuxFormat



Don't panic! This is standard behaviour: a grey startup screen.

wayland-99. The actual invocation then starts the display server – it displays a grey startup window similar to the one shown in the screenshot *(above)*.

As we appended the ampersand symbol to the back of the `ubuntu-frame&` command, the terminal does not block while Ubuntu Frame runs. However, status messages (of which there will be many) emitted by the *Ubuntu Frame* subsystem will show up in the command-line window along with any output caused by programs invoked later.

As long as the variable is set to the target display, any invoked Wayland application shows up in it.

A nice and Canonical-sanctioned demonstration for this involves the *Mastermind* application. Download and launch it via the following two commands:

```
$ sudo apt install gnome-mastermind
$ gnome-mastermind
```

After the application starts up, the *Ubuntu Frame* instance displays the status message `[2023-07-08 20:25:02.146823] <information> frame: New surface with title="GNOME Mastermind"` . This confirms the successful connection between the application and its display server.

Adapting applications to run in the *Ubuntu Frame* subsystem is dependent on the framework used by the application developer. Some *Qt*-based programs, for example, need an extra invitation before enabling Wayland connectivity.

Fortunately, Canonical provides a whitepaper at **https://bit.ly/lxf306paper.** It describes approaches needed to get commonly used frameworks working with a Wayland server.

QUICK TIP

The process outlined in this article is frugal on computer resources. Do not hesitate to use process computers that, otherwise, can no longer be used to power your smart displays – overprovisioning is not necessary or economically sensible.

Smart infrastructure

After erecting an *Ubuntu Frame*-based test harness on the workstation, it is time to create some smart display architecture. For that, we will use the services of *Dashkiosk* found at **https://dashkiosk.readthedocs.io/en/latest/#** – while its administrator has abandoned the product, Canonical recommends it in official documents. Furthermore, the long-running product has matured and is available as a ready-to-deploy Snap package. The actual architecture of the product is shown in the screenshot *(below)*.

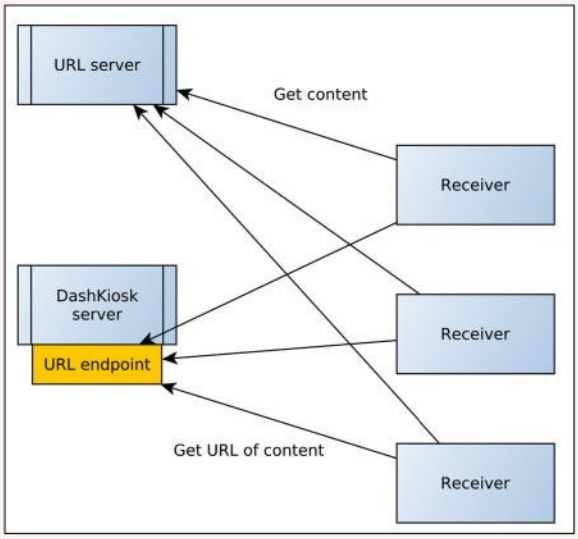
The *Dashkiosk* ecosystem provides a range of additional apps, such as Android and Chromecast versions of the product, but we're not using them here.

Now return to the main workstation, where we use `snap install` to deploy the *Dashkiosk* server:

```
$ sudo snap install dashkiosk
```

At the time of writing, running the above-mentioned commands yield the message `dashkiosk v2.7.11 from Oliver Grawert (ogra) installed` – it is possible that the Snap package will be updated before the time this article reaches your hands.

Interestingly, the *Dashkiosk* application does not integrate itself into the command line. Instead, the



A server is provided in the centre that feeds the individual client components, which are called receivers, with displayable material.

» DISADVANTAGES OF CENTRALISATION

While the process shown here greatly simplifies maintenance of the cluster because the special offers and ads to be displayed can be managed in a central location, be aware that there are disadvantages of this method.

If the central terminal fails, all of the satellite displays are also taken offline. Fortunately, this problem can be addressed in multiple ways. First of all, using a higher-quality computer (such as a used Dell OptiPlex or similar workstation) limits the number of potential failures – using a bargain-basement computer in such an application is, obviously, not the smartest of ideas.

Secondly, the relatively low demands on compute are helpful because they mean that a hot spare can be purchased for relatively little money – doing so makes reacting to an emergency situation easier, because even an unskilled worker can be instructed to power off one machine and switch on another in its place.

An extra problem crops up if the systems are not on the same local area network – network operators can and do have network outages, with sometimes disastrous results.

Selecting the best approach is clearly a question of your individual situation. Do, however, keep the problems outlined here in mind before committing to an overly centralised solution.

service starts up automatically – open a browser window and go to **http://127.0.0.1:9400/admin#/groups** to reveal the administration interface used for receiver management.

Equally interesting is **http://127.0.0.1:9400/receiver**, which contains the receiver interface. Invoke it to see a series of nature photographs selected by the developers of *Dashkiosk*.

Assuming that the Ubuntu Core-based Raspberry Pi and the main workstation are on the same network, we can proceed to creating the receiver instance on the process computer.

First, switch to the command-line terminal connected to the Pi, and enter the following command to download the *Ubuntu Frame* display server and the WebKit component intended for kiosk displays:

```
$ sudo snap install ubuntu-frame wpe-webkit-mir-kiosk
```

To recapitulate and clear the verbiage: standalone smart display applications have, since the beginning of time, also been called Kiosk Mode. A legacy of this can be found in smartphones, for example – the display mode intended to be used in stores to attract customers is usually called kiosk mode.

Beginners are often confused as to why the *Dashkiosk* package is not installed on the receivers. A careful look at the architecture screenshot *(left)* reveals that the server provides URLs to its clients – they use a web browser of choice for display.

Be that as it may, the Raspberry Pi will start to display advertising for the WebKit browser once the install command has run its course. This is normal, and represents a sort of attract mode for the downloaded WebKit component.

The next step involves pointing the package at the server. This can be done by using the following command, which modifies a configuration parameter exposed by the Snap package:

```
$ sudo snap set wpe-webkit-mir-kiosk
url=http://192.168.1.68:9400/receiver
```

After entering it, a few seconds have to pass – after that, the sample images contained in the distribution show up on the display attached to the Raspberry Pi. Of course, the command shown needs to be adjusted to your local network situation – our workstation lives at the IP address **http://192.168.1.68**.

Advanced configuration

While showing pictures of animals and landscapes is a nice touch for interior design, the original goal of this article was the creation of a smart display system intended to convey advertising, special offers or other information of commercially valuable nature.

To achieve this goal, multiple avenues of configuration are available. The first level is done via the file **/var/snap/dashkiosk/current/config.json** – by default, its configuration looks like the one shown in the configuration screenshot *(above-right)*.

This file can be edited to modify system behaviour – for example, the port variable can be changed in order to make the server expose its interfaces using a different TCP port.

In addition to that, various extra configuration options are available– further information on the topic can be found by visiting the documentation at **https://dashkiosk.readthedocs.io/en/latest/configuration.html#json-configuration-file**.

```
tamhan@TAMHAN18:~$ cat /var/snap/dashkiosk/current/config.json
{
    "environment": "production",
    "port": "9400",
    "db": {
        "options": {
            "database": "dashkiosk",
            "storage": "/var/snap/dashkiosk/10/db/dashkiosk.sqlite",
            "dialect": "sqlite"
        }
    },
    "log": {
        "file": "/var/snap/dashkiosk/10/dashkiosk.log"
    }
}
tamhan@TAMHAN18:~$
```

Here's the default JSON configuration file with required port, logging file and database.

An interesting aspect is the aforementioned configuration interface – open it after the Raspberry Pi connects to reveal two displays as shown *(below)*.

This interface groups the smart displays, thereby permitting targeting of the individual creatives.

Modifying the content shown on the displays is more difficult, however – the *Dashkiosk* framework limits itself to assigning a display URL and does not provide rendering except for the basic default image presenter, which can't be customised. Assigning different content requires you to open the *Dashkiosk* configuration interface in the first step, where one or more groups of dashboards must be created. These can be assigned a URL pointing to the content provider – in addition to classic dashboard applications such as *Grafana*, you are free to use any classic web-based image display system or even a custom JavaScript application of choice.

Automatic startup

In some cases, automatic startup of the infrastructure is not ensured. This is especially annoying if power outages are a regular occurrence, as non-technically minded workers can struggle with command-line tasks.

Should the Raspberry Pi at hand be unwilling to start the kiosk mode automatically after a power outage, enter the following three commands:

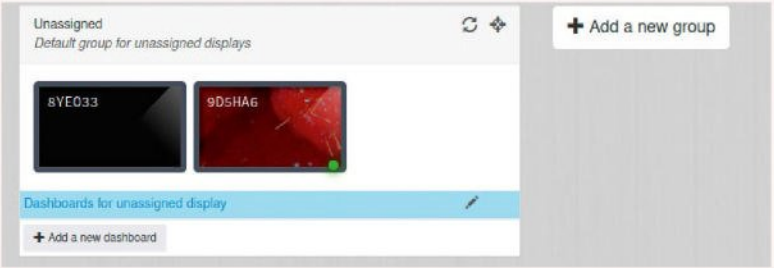
```
$ sudo snap set ubuntu-frame daemon=true
$ sudo snap set wpe-webkit-mir-kiosk daemon=true
$ sudo snap set wpe-webkit-mir-kiosk url=http://
localhost:9400/receiver
```

Setting the **daemon** property to **true** informs Ubuntu Core that these services must launch during system startup – when set, the dashboards appear automatically instead of the configuration message.

The combination of *Ubuntu Frame* and *Dashkiosk* allows for flexible smart displays using open source systems. Once the structure outlined in this tutorial works, any classic dashboard framework can be used to generate content. LXF

QUICK TIP

If the power supply of the process computer fails, the system is offline. If you want high reliability, be careful not to pick bargain-basement parts when designing the smart display system.



The default display configuration is set for dual display.

SAFECLOSET

Credit: https://github.com/Canop/safecloset

Best kept secrets

A man who keeps his cards close to his chest, we struggled to persuade **Shashank Sharma** to reveal how he protects his passwords.

OUR EXPERT

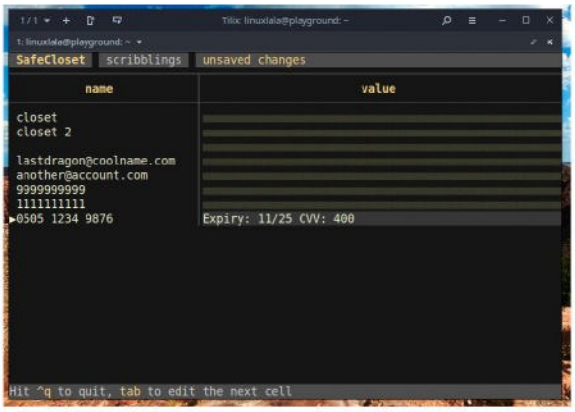
Shashank Sharma is a trial lawyer in New Delhi, India, and an avid Arch user. He's been writing about open source software for 20 years and lawyering for 10.

Everybody's got something to hide except me and my monkey, said the Beatles back in the day. However, the correct phrase for the modern world would be, "Everybody's got something to hide, especially me and my monkey." If you're still keeping all your passwords and other important information in a draft email, *SafeCloset* is for you.

One of the first mistakes people make is using the same username and password combination across different online services because of their inability and unwillingness, understandably, to remember multiple login credentials. With *SafeCloset,* you can store all your different login credentials and other important information in secure, password-protected files. While there are many password managers on offer, some of them built into modern desktop browsers, the advantage of having all your secrets at your fingertips, rather than surrendered to the safety of a third party service, cannot be overstated.

You'll have to check the project's GitHub page or the website (**https://dystroy.org/safecloset**) for a complete list of features. For now, you'll appreciate that the project is cross-platform, which makes it possible for you to access your secrets from Windows, Mac and Android devices, in addition to all flavours of Linux .

Written in Rust and released under the AGPL-3 licence, the project isn't available in the software repositories of popular desktop distros. If you don't already have Rust and *Cargo* installed, you can use your distro's package manager to install these: `sudo dnf install rust cargo` installs them on RPM-based distros such as Fedora. You can similarly run `sudo apt install cargo` , which also installs the **Rustc** package, if you're running Ubuntu or Debian, or a derivative.



While you can't use SafeCloset to secure files, you can secure any alpha-numeric data with it.

You can now easily install *SafeCloset* from the source. First, clone the GitHub repository with `git clone https://github.com/Canop/safecloset.git` , then navigate into the **safecloset/** directory and run `cargo install --locked --path .`

Alternatively, you can download the ZIP file from the Releases section. This comprises multiple binaries –

» MIND YOUR DRAWERS

When adding entries to a drawer, you can press the Tab key after adding the value to create a new entry, or press Enter and then `n` to create a new entry in the current drawer.

SafeCloset's top bar is deliberately bare so you can see whether there are any unsaved changes in the drawer. To save a drawer, such as after creating a new entry, press Ctrl+s.

Press Ctrl+u to close the currently open drawer. *SafeCloset* automatically saves changes in the drawer before closing it.

If you need to edit any entry in your drawer, select it from the list and press `a` . You can reorder entries in a drawer using Ctrl+Up Arrow or Ctrl+Down Arrow. When adding the value for an entry in a drawer, press Alt+Enter to create a new line.

You can also use the menu, which is accessed by pressing Esc, to select the operation you wish to perform, including changing the password for the drawer. It's also possible to import data from a drawer in the current closet or from a drawer in another closet.

SafeCloset will not create an empty closet file. So, in our example, the **~/Documents/scribblings** file is not created until after we create a drawer and save some new entries within it. It's good practice to copy this closet file and keep it as an added safety measure.

move the suitable **safecloset** binary to a directory in your $PATH. In our case, we moved the **build/x86_64-linux/safecloset** file to the **~/.local/bin** directory.

Closets and their secrets

To begin using *SafeCloset*, you first have to create a closet. You can create as many closets as you like, but you have to create at least one drawer for each. The actual secrets, or information, are stored within such drawers. Both closets and their comprising drawers are password-protected.

SafeCloset also allows you to create nested drawers, but the official documentation warns against using drawers as categories. As all closets and drawers require individual passwords, carefully examine your needs before creating closets and drawers.

You launch *SafeCloset* with `safecloset <directory>/<closetname>`. If the closet already exists, *SafeCloset* opens it, otherwise it creates a new one. For instance, the command `safecloset ~/Documents/scribblings` creates a closet called **scribblings** within the **~/Documents** directory.

When you create a new closet, *SafeCloset* informs you that this is a new closet and that you can press Ctrl+n, written as ^n on the screen and in the project's documentation, to create a new drawer. You're asked to provide a passphrase for the new top-level drawer. You can choose any password of any length, or even opt for a passphrase if you so desire.

You'll notice the name of the closet on the top bar. *SafeCloset* doesn't allow you to provide names for the drawers, and the interface doesn't list how many drawers there are within any given closet. This is deliberate, to help protect your sensitive information. While you can provide any name you like for your closet, the project doesn't let you name the individual drawers within it. The drawers are only identified by the password or passphrase used to unlock them.

To open a drawer, press Ctrl+o and enter the passphrase or password you used to safeguard the drawer. If you have created nested drawers, and wish to work with a deep-level drawer, you have to first open all the parent drawers.

With a drawer now open in the current closet, you can finally start relegating important information to *SafeCloset*. To begin, press `n`.

The *SafeCloset* interface now splits into two panes. On the left, you provide a name to identify the information you wish to store, and on the right pane the actual information to be safeguarded.

Name	Value
closet 2	Thisisthepassword
closet 3	When working with several closets
closet 4	Probably a good idea to store passwords in a secure closet
5	Passwords and passphrases can have spaces

By default, all the secrets, stored in the value field, are visible. You can press Ctrl+h to hide them. You navigate through the entries in a drawer using the up and down arrow keys. To view the associated secret for an entry in the drawer, press Tab and the hidden value is displayed, while values for other entries stay hidden.

If you want to always keep the passwords, passphrases and the values for the entries in a drawer hidden, you can start *SafeCloset* with the **-h** command switch. The command `safecloset ~/Documents/scribblings -h` opens the **scribblings** closet but the password you type to unlock the drawer and the actual values in the drawer are hidden.

Searching for answers

There's no limit to the number of entries you create within a drawer. If you end up with hundreds of entries in a closet, looking for information can be quite a pain.

Thankfully, *SafeCloset* offers a powerful search feature. Press `/` to invoke the search and type in the keywords to identify the entry you're interested in. To match entries in the drawer, *SafeCloset* looks only in the name field, not the value field. It also ignores case and diacritics when performing a search, so searches for `Twitter` and `twitter` yield the same result.

When multiple results are displayed, you can use the up and down arrow keys to navigate through the results and press Enter to select the one you want.

To access help, press `?` from the *SafeCloset* interface. This is quite thorough and covers all the options available to use within *SafeCloset*.

There's a number of operations only available from within a drawer, such as opening, creating, saving and closing a drawer, changing a drawer password, and so on. You can access all of these from a menu within the *SafeCloset* interface by pressing Esc from inside an open drawer.

To close a search, you must press `/` again and then Esc. If you directly press Esc, *SafeCloset* opens the menu instead of closing the search. To exit *SafeCloset*, select the option from the menu or press Ctrl+q.

Unlike most other text utilities, *SafeCloset* doesn't allow any operation or tweaking to be done by editing config files or using command options. Forcing users to perform all operations from within *SafeCloset's* TUI is yet another deliberate feature.

After having used a number of password managers over the years, this author can confidently state that *SafeCloset* is his favourite. LXF

As an added security feature, SafeCloset automatically closes after 120 seconds of inactivity.

QUICK TIP

If no <path> is provided to the `safecloset` command, the specified closet would be created in the current working directory. You can end up with multiple closets with the same name across different directories if you are not careful.

» **ENHANCE YOUR TERMINAL-FU** Subscribe now at **http://bit.ly/LinuxFormat**

STEAM DECK

Upgrade your Steam Deck SSD

Neil Mohr asked hardcore PC gamers **Tony Polanco** and **Katie Wickens** to upgrade a Linux device – what could possibly go wrong?

OUR EXPERT

Neil Mohr only gets to see his Steam Deck once in a while through the sweaty grip of a six-year-old playing Lego Batman.

Before we start, a warning: opening your Steam Deck and upgrading it invalidates your warranty. You do so entirely at your own risk. *Linux Format* cannot be held responsible for any damage – or frankly anything ever. On a less legalese level, it is a good idea to wait until your warranty has expired, but otherwise, let's continue!

The Valve Steam Deck has transformed Linux into a handheld gaming success story. While the base model is an affordable £350, its 64GB storage is easily swamped with even vaguely modern game installs. Sure, you can expand it via the microSD slot, but its speeds are much slower than internal storage.

With solid-state storage costs plummeting, the good news is that if you don't mind opening up your Steam Deck (and invalidating any warranty), you can pretty easily swap the internal storage for a 256GB NMVe 2230 drive for as little as £30, quadrupling your storage. Many people go for 1TB but that's about £100.

The Valve Steam Deck has become something of a Linux-powered phenomenon.

Upgrading the Steam Deck SSD might seem scary, but it doesn't have to put you into a spiralling panic, or even be super-time-consuming. Before you embark on

YOU NEED

- Steam Deck
- 2230 NVMe SSD
- Tweezers, plectrum or pry tool
- PH1/0 screwdriver
- 8GB USB-C drive (or USB-C adaptor)
- NMVe adaptor (clone option)

» PERFORMANCE OVERLAY

Knowing how to enable the performance overlay on the Steam Deck helps you get more from it immediately. Astonishingly, given it's a handheld, the Steam Deck has all of the functionality of the desktop version of Valve's game launcher and marketplace. That means you can enable the performance overlay to see a game's frame rate. This is important if you want to see how well a game performs at different graphical settings. The Steam Deck also provides additional metrics that could be useful, and they can all help you figure out whether you've got the flexibility to tweak settings to maximise performance or battery life.

Turning on the performance overlay is easier on Steam Deck than on the desktop version. That's because you can easily bring it up by pressing the Quick Access Menu (QAM) button. Within the Performance section, you'll see a slider with four levels. Moving to a higher level shows more performance metrics. This neat feature does an excellent job of displaying how well (or poorly) a game runs on Steam Deck.

Click the QAM button, which is located on the right-hand side of the screen and beneath the right touchpad. Within the QAM, you'll see five different tabs. Scroll down to the battery icon, which represents the Performance tab, and press A to enter this menu. You'll find the performance overlay level beneath the battery life counter. Head to Performance Overlay Level and press A to enter this submenu.

You'll see four performance overlay levels along a slider. Move the slider right or left to select different levels. The first shows the current frames per second. Higher levels display things such as GPU and CPU performance percentages, fan speed and more.

Click on Advanced View to see a list of additional settings. You can set frame rate limit, refresh rate, thermal power (TDP) limit, manual GPU clock control and more. It can get pretty granular, but this is one of Steam Deck's main selling points – it's extremely customisable. Click the B button a couple of times to hide the QAM screen to exit. To disable metrics, just follow the previous steps.

a Steam Deck SSD upgrade, the first step is figuring out whether you want to reimage your Steam Deck (easy) and start from scratch, or whether you need to fully clone (more involved) your Deck.

If you only have a few games on your Steam Deck and aren't on a metered internet connection, it's probably smarter just to reimage the Deck with the latest SteamOS, and maybe copy your game files over using Steam's new, default file transfer service. It also skips the need to buy an NMVe adaptor, which could cost as much as the new NMVe SSD.

If you've spent hours upon hours tweaking your settings in Desktop mode, or have a whole bunch of emulators and ROMs, for example, a SteamOS reimage is going to wipe all your hard tinkering away. Instead, you'll want to clone the Steam Deck SSD, to avoid the hassle of redownloading, switching settings and navigating the Linux-based maze that is SteamOS.

We take you step-by-step through a straight replace and imaging in the walkthrough below, but we'll outline how you can image and backup your existing SteamOS install if you read on. This requires an NMVe external adaptor, which typically costs around £20, but can be useful in this day and age, so you might be happy picking one up so you can reuse your Deck's old internal drive, for example.

Becoming more common now, single-sided NVMe M.2 2230 SSDs used to be as rare as hen's teeth.

Deck to deck

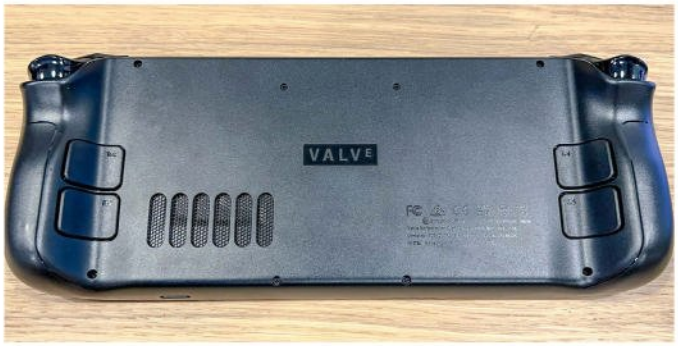
Before starting, we suggest you format your new SSD beforehand as a GPT-partitioned disk. The format doesn't matter because it'll be imaged over, but it needs to be GPT-partitioned for the Steam Deck to boot. You can do this on any system.

On the Deck, remember to disable (or just turn off) Family mode if you use it, because this disables Steam Desktop access as an option (which is very confusing if you don't realise). Plus, to access the on-screen keyboard in Desktop mode, the Steam app needs to be

REMOVE AND UPGRADE YOUR DECK'S SSD

1 Battery Storage mode

Before opening, Valve recommends enabling Battery Storage mode for safety; the Steam Deck won't respond to power button presses. Plug the Deck into the charger and power it down. Hold down the volume-up (+) button and press the power button. Release both buttons. Using the D-Pad and A button, select Setup Utility > Power Menu > Battery Storage Mode.



2 Unscrew the Steam Deck

Unscrew the eight screws on the back of the Steam Deck. The four inner screws are shorter while the four outer screws are longer. Remembering this helps when putting the system back together. Use a seam splitter or a sturdy piece of plastic to open the Steam Deck. Don't use anything metal because you want to avoid damaging the handheld.

3 Remove the inner metal sheet

Next, you have to remove the metal sheet covering the SSD. This has up to four screws – ours had three. There's one screw on the top-left and two screws on the bottom. You'll find another screw hidden beneath a sticker on the upper-right. Peel the sticker back to expose the screw but don't completely remove the sticker because you have to put it back down later.

4 Unplug the battery?

As we're using Battery Storage mode, this step is optional as it won't accidentally power on, but for extra safety, you can unplug the battery. It is slotted in tightly but you should remove it by hand instead of with a tool. Use your fingernail or the edge of your pry tool to slowly pull the battery out. Don't rush this part because you could easily damage the battery.

QUICK TIP

Ensure you format your SSD beforehand as a GPT-partitioned disk; the format doesn't matter as it'll be imaged over, but it needs to be GPT-partitioned for the Steam Deck to boot.

running. Accessing the on-screen keyboard is done by pressing Steam+X.

Now hold the power button when the Deck is turned on and tap Switch To Desktop Mode. You might want to set a sudo password before you continue as it will ask for one later, so go to the bottom bar and tap the app launcher (bottom-left).

Then either go System > Konsole and type `passwd` to set a password, or click your Steam Deck user profile image at the top of the app launcher, and hit Change Password on the right. Pick something simple, because when you type it in the Konsole, you won't be able to see your password at all. It doesn't even come up with asterisks as you type, so don't choose a 20-character-long monster password. Now type `$ lsblk` in the Konsole.

Hit Enter and it probably asks for your password at this point. Type that in and hit Enter, and it lists all your storage devices. You want to make a note of your current NVMe drive's name. Usually that's something along the lines of **nvme0n1**.

A wise buy enables you to reuse NVMe drives of any size; this adaptor cost £12.

Make sure the Deck isn't flagging, battery-wise, and plug your new NVMe drive into the Steam Deck using your adaptor. Type the following into the Konsole – but do check **nvme0n1** and **sda** against your own settings:

```
$ sudo dd if=/dev/nvme0n1 of=/dev/sda
oflag=sync bs=128M status=progress
```

Hit Enter and it should copy everything over. Depending on how many games you have downloaded, it might take a while. Then just follow steps one to five of the walkthrough to swap out the old NMVe SSD. Plug it in to bring it out of Battery Storage mode and, hey presto, it should boot, no questions asked. Check out your storage situation and gloat at beating the system! LXF

REMOVE AND UPGRADE YOUR DECK'S SSD *CONTINUED*

5 Replace the SSD

Undo the screw holding the SSD in place, remove the SSD's shield covering, then pull the SSD out. Place the shield over the SSD you want to install. Insert the new SSD and screw it into place. The final part of the physical process is to put your Deck back together. Plug the battery back in, screw the metal sheet on, put the back cover on and screw the eight screws in.

6 Download SteamOS

Now we need to reinstall SteamOS. On your PC, head to **https://bit.ly/lxf306steamos** and click Download SteamOS Deck Image. This ISO image needs to be written to the GPT-formatted USB stick. To keep life simple, you can grab *Etcher* from **https://etcher.io**, install that and get it to write the Steam Deck ISO file to the USB stick.

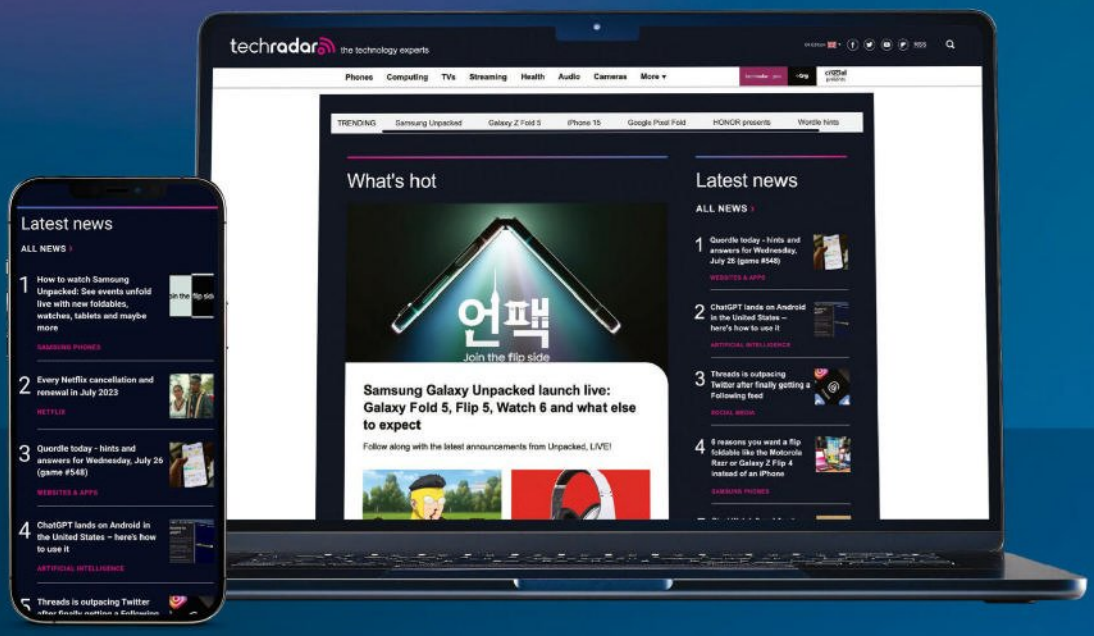
7 SteamOS on Deck!

Plug the USB stick into the Deck. Press and hold the power and volume-down buttons. At the startup beep, release the power button but keep holding the volume-down button until the BIOS appears. Select your USB device and it should boot to the desktop – we had to reboot once. Click Reimage Steam Deck from the Desktop mode and follow the on-screen instructions.

8 Enjoy the new storage!

The Steam Deck first has to initialise the new drive, but once in, you'll find you have a whole universe of new space under Storage. If you drop to Desktop and install *KDiskMark*, you can even test your drive. The old external microSD probably ran at 100MB/s read, while our 512GB NMVe runs at almost 3,000MB/s – that's 10 times faster than the original internal storage.

Meet the technology experts


techradar
the technology experts
Phones
Computing
TVs
Streaming
Health
Audio
Cameras
More
TRENDING
Samsung Unpacked
Galaxy Z Fold 5
iPhone 15
Google Pixel Fold
HONOR presents
Wordle hints
What's hot
Latest news
ALL NEWS
Join the flip side
Samsung Galaxy Unpacked launch live: Galaxy Fold 5, Flip 5, Watch 6 and what else to expect
Follow along with the latest announcements from Unpacked, LIVE!
1 Quordle today - hints and answers for Wednesday, July 25 (game #548)
2 ChatGPT lands on Android in the United States – here's how to use it
3 Threads is outpacing Twitter after finally getting a Following feed
4 6 reasons you want a flip foldable like the Motorola Razr or Galaxy Z Flip 4 instead of an iPhone
Latest news
ALL NEWS
1 How to watch Samsung Unpacked: See events unfold live with new foldables, watches, tablets and maybe more
2 Every Netflix cancellation and renewal in July 2023
3 Quordle today - hints and answers for Wednesday, July 26 (game #548)
4 ChatGPT lands on Android in the United States – here's how to use it

- The world's **most** comprehensive technology website
- An **unrivalled** mix of news, opinions, reviews and features
- **All-new design**, new homepage, new features and special reports
- Backed by **over 300 years** of editorial experience

techradar
the technology experts

www.techradar.com

NGINX

Credit: http://nginx.org

Access services with Nginx reverse proxy

Nick Peers discovers how to open your network services to the internet with this user-friendly implementation of Nginx.

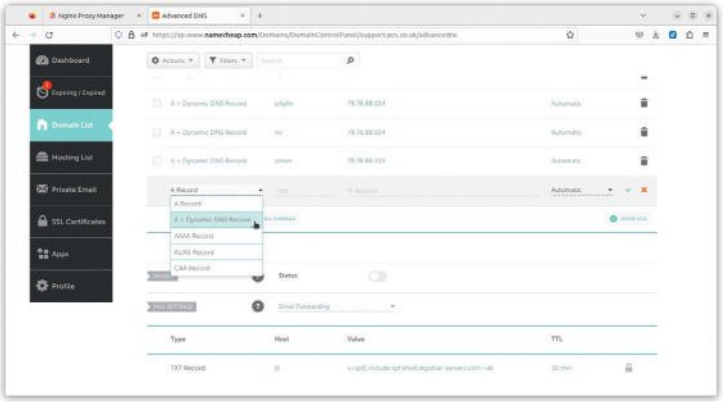
OUR EXPERT

Nick Peers Is obsessed with self-hosting. From Bitwarden and Jellyfin to Nextcloud and MotionEye, all his services are so much easier to configure for remote access with *Nginx Proxy Manager*. He is the life of the party, too!

If you're looking for a way to access self-hosted services from outside your home, you have two basic choices. For maximum security, you'd set up a VPN tunnel that you'd need to dial into every time you wanted access, but if you'd like a simpler option that's still secure, and gives others access, you want a reverse proxy.

One of the best-known tools for the job is *Nginx* (**https://nginx.org**), but it's not the most user-friendly, even when wrapped up in deployment-friendly containers like Linuxserver's *Swag*. If the idea of messing around with separate config files for each service you want to open up sounds too much like hard work, you'll love *Nginx Proxy Manager*. It enables you to set up, view and administer all your connections via a pleasingly easy-to-use web front-end.

Like *Swag, Nginix Proxy Manager* is distributed as a *Docker* container. We're assuming you have *Docker* set up and configured, and have some knowledge of how it works. *Nginx Proxy Manager* can be installed on any machine on your network, but we're assuming you'll want it running 24-7, so you'll want it on your dedicated server where it's likely most other shared services are also running, many of which may be running in *Docker*.



Creating subdomains for each of your services makes them easier to configure for remote access using Nginx Proxy Manager.

Nginx Proxy Manager can redirect traffic anywhere on your network, but if you want to direct traffic to other *Docker* containers, they must all be on the same (custom) bridge network as *Nginx Proxy Manager*. If you've not yet set up this shared bridge, it can be done with a single *Docker* command, substituting `shared-net` with your choice of shared network name:

```
$ docker network create shared-net
```

You need to then connect your existing *Docker* containers to this network – if you're administering them through the *Portainer* web interface, this is simple enough. Navigate to the container's details page, scroll down to Connected Networks, where you can select your new network and click Join Network.

If you're managing *Docker* from the command line, you need to stop and remove each container in turn, then insert the following line into the setup script you use to recreate it:

```
--net=bridge-for-all-seasons \
```

Get your domains in order

You've configured *Docker*, but before you install *Nginx Proxy Manager,* there's a couple more prerequisites to

» INSTALL NGINIX PROXY MANAGER

The quickest and easiest way to deploy *Docker* containers is by pasting the setup script into a text editor, then saving the file in plain text so you can easily copy and paste it into a terminal window, plus make edits whenever required.

The script here, of course, needs adapting to your personal setup, namely the name of your shared network under `--net` and the location of your config and LetsEncrypt certificate folders under `-v` . You may also need to adapt the `-p 81:81 \` line if port 81 is already in use (for example, change it to `-p 82:81` to access *Nginx Proxy Manager* through **http://192.168.x.y:82**).

```
docker run -d \
  --name=nginx-proxy-manager \
  --net=shared-net \
  -p 443:443 \
  -p 80:80 \
  -p 81:81 \
  -v /home/dockeruser/
containers/nginx/data:/data \
  -v /home/dockeruser/
containers/nginx/letsencrypt:/
letsencrypt \
  --restart unless-stopped \
  jc21/nginx-proxy-
manager:latest
```

satisfy. The first is that you need access to a suitable domain to provide access to your servers from the outside world – this can be your own domain, from which you can optionally create separate subdomains for each service you have, such as **jellyfin.domain.com**, or you could make use of a dynamic domain like that offered by NoIP (**www.noip.com**).

If using your own domain, start by making a note of your current public IP address (use **www.whatsmyip.com**), then log into your domain provider and make sure the domain (plus any relevant subdomains if applicable) are pointing to the same IP address.

Unless this is a fixed IP address, you also need to install a DNS updater, so when your ISP changes your IP address, your domains are automatically redirected. Check your domain provider – many support *DDClient* (**https://ddclient.net/protocols.html**), either as a standalone tool or via the Linuxserver *Docker* instance (**https://docs.linuxserver.io/images/docker-ddclient**).

With your domains set up and pointing towards your home, it's time to configure your router to redirect all web requests to the server you'll install *Nginx Proxy Manager on.* Consult your router's documentation for full details, but look in the port forwarding section and direct traffic on both port 80 (http) and 443 (https) to your server's private IP address (**192.168.x.y**). This reveals another benefit of a proxy – you don't need to configure lots of ports in your router; simply funnel everything through standard web ports to your server, then let *Nginx Proxy Manager* distribute them.

Nginx Proxy Manager

It's time to add *Nginx Proxy Manager* to your server. The install box *(opposite page)* reveals the script you need to adapt to your own setup, but first you need to create two folders, inside which you'll store your *Nginx* configuration data and SSL certificates – in the example script, we've set up an **nginx** parent folder, inside which are two subfolders: **data** (for **config**) and **letsencrypt** (for the SSL certificates).

A quick summary of what the script does: it pulls and installs the latest version of *Nginx Proxy Manager,* joins it to the shared network you set up earlier and redirects three outside ports into Nginx: 80 and 443, as we've already explained, while port 81 is used to connect to its web-based interface.

If you've set things up correctly, the container starts without a hitch and should be available within seconds.

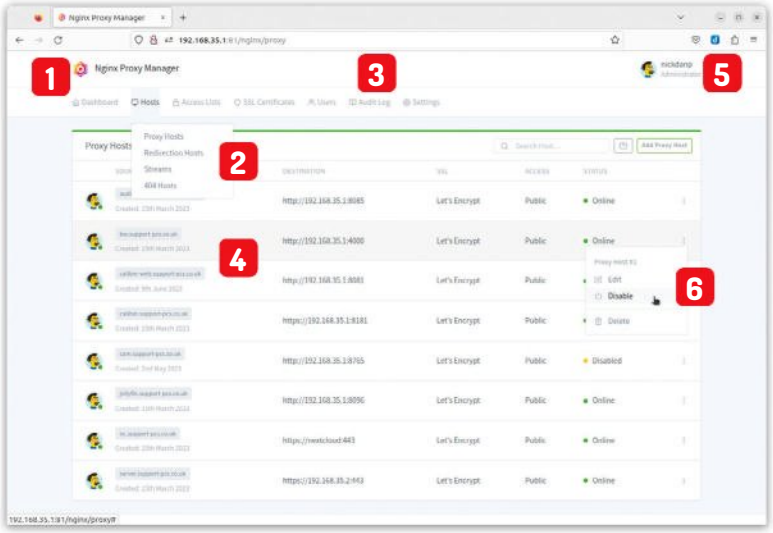
Browser-based access

Once installed, *Nginx Proxy Manager* can be accessed through any device on your network via any web browser. Simply navigate to **http://192.168.x.y:81**, substituting **192.168.x.y** with your server's IP address, which takes you to a login screen.

Log in using **admin@example.com** as your username, and **changeme** as your password. You'll then be prompted to change the email address (as well as provide a name and nickname). Click Save to change the password – make it a secure password generated by (and stored in) your password manager, even if you only intend accessing *Nginx Proxy Manager* locally.

Once done, you'll find yourself at the Users screen. Above this you'll see there are seven distinct sections to navigate – start by clicking Dashboard for an overview. Here you'll see *Nginx Proxy Manager* supports four types of proxy: Proxy Hosts, Redirection Hosts, Streams and 404 Hosts.

NGINX PROXY MANAGER OVERVIEW



1 Overview
Click on the Dashboard tab to get a quick overview of how many proxies you have set up.

2 Hosts
Nginx Proxy Manager enables you to set up one of four different types of proxy.

3 Audit log
Click here to get a complete – and detailed – list of all the operations and changes you've performed.

4 View summary
Get an overview of each proxy host with this handy and easy-to-read list.

5 User access
Click Users to give others access – this can be read-only or administrator, depending on your level of trust.

6 Quick actions
Click the vertical ellipsis to make changes to existing proxies, plus disable them temporarily or delete them.

Your very first proxy

Let's start with Proxy Hosts. The step-by-step guide *(next page)* reveals how to set up a simple proxy host, which enables you to redirect traffic from a specific domain, subdomain or dynamic domain to your choice of IP address and port. When choosing the scheme (http or https), stick with whatever you use to connect locally – for example, the self-hosted Bitwarden *Docker* instance Vaultwarden only uses http. Don't worry, this doesn't mean your connection is insecure – that's configured separately.

You have a choice between hostname and IP address of the machine you're redirecting to – if your service is on a machine with a static IP address, use that; if it's on machine with an IP address allocated to it by your router, try the hostname route. If that doesn't work, tie that machine down to a static IP address.

When bad actors try to attack websites, they often use rather basic techniques, such as attempting to inject malicious text into insecure web forms. When you open your servers up to internet access, you put them at risk from such attacks, which could have wider consequences for your network.

Nginx Proxy Manager offers a Block Common Exploits switch for all proxy and redirection hosts, which offers protection against the most common forms of attack. These are listed in the **block-exploits.conf file**, which can be found by navigating to **https://github.com/NginxProxyManager/nginx-proxy-manager**

QUICK TIP

If you're unable to get Access Lists working the way you want them to, consider examining other possible solutions – for example, your router's firewall might offer a way of blocking unwanted traffic from known bad actors or restricting access to IP addresses from a specific part of the world.

QUICK TIP

Select Users on the *Nginx Proxy Manager* dashboard to give others access to *Nginx*. You can give them view-only access to your proxies or give them full administrative access – ostensibly for their own services, but they also have full access to yours, so be warned.

and then drilling down to **docker/rootds/etc/nginx/conf.d/include**. You'll see it's a simple script that intercepts both SQL and file injections, as well as common exploits, spam and user agents.

If you're using a single domain with subfolders, as outlined in step three of the walkthrough, you can configure all those subfolders in a single proxy host. You still need to fill in redirection details for the parent domain on the Details tab, so consider where you'd like people entering that address to be redirected.

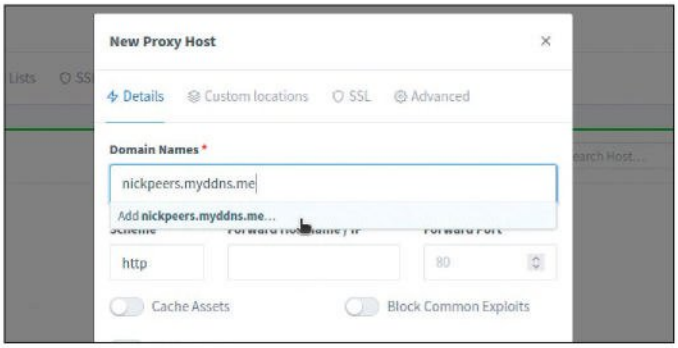
When you come to secure outside web connections to your server, you need an SSL certificate. While you can provide this yourself, *Nginx Proxy Manager* has built-in support for requesting free certificates from LetsEncrypt. Not only does it configure these automatically, but it also handles the otherwise arduous chore of remembering to renew the certificate every three months. Once added, you can view (and manage) your SSL certificates via their own dedicated section – you get a handy summary of each, including the certificate provider, its expiry date and options to renew manually now, download the certificate to your PC, test the server reachability and remove if you have no further use for it.

Adding user credentials

Opening local services to access from outside your network comes with obvious risks, only some of which can be mitigated using the switch to block common exploits. While some services may have built-in authentication (for example, Nextcloud or Jellyfin), others may give anyone with knowledge of your domain name unfettered access. *Nginx Proxy Manager's* Access Lists offer you two ways to mitigate this: simple http authorisation, which forces anyone visiting the site to enter a username and password before they can proceed, and a combination of whitelists and blacklists, enabling you to restrict access to specific clients only.

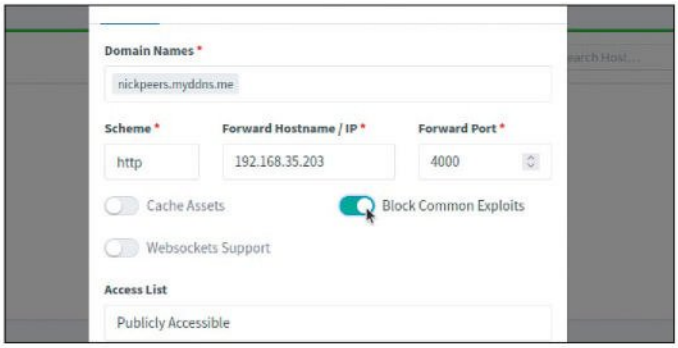
Here's a simple example that uses http authorisation. Select Access Lists in *Nginx Proxy Manager,* then Add Access List to get started. Give

SET UP A PROXY HOST



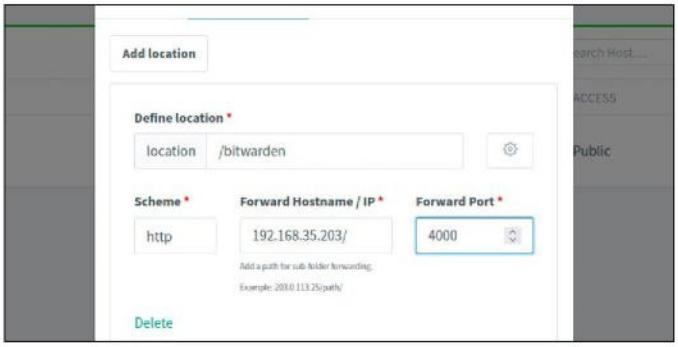
1 Configure basic settings

Click X Proxy Hosts followed by Add Proxy Host. The New Proxy Host window opens to show four tabs. Start by entering the domain, subdomain or dynamic domain you'd like to use for this proxy into the Domain Names field, clicking Add when done. Next, enter the details (scheme, host and port) of where you're redirecting the domain.



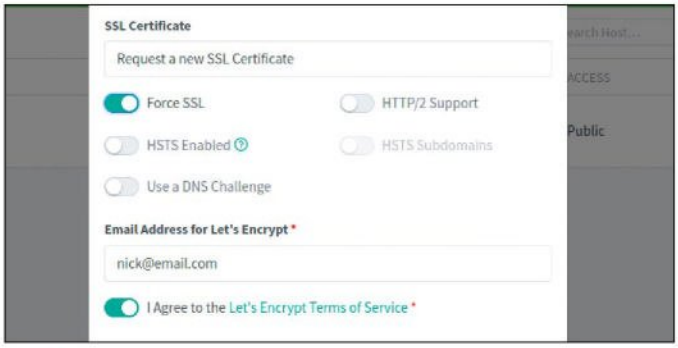
2 Set optional switches

There are three more switches: Cache Assets can speed things up if the service is on a different computer from *Nginx Proxy Manager* and pulls in lots of static content. Block Common Exploits is discussed in the main copy – it's worth switching on for security reasons. Leave Websockets Support disabled until you've tested the connection; try enabling it if it doesn't work.



3 Define custom locations

If you want to connect to a domain using subfolders (such as **https://domain.com/bitwarden**), switch to Custom Locations and click Add Location to fill in the details – it's self-explanatory: choose a path name, then select the scheme, hostname/IP address and forwarding port as in step two. Leave a **/** after the hostname or IP address (such as **192.168.x.y/**) or it won't work.



4 Add SSL certificate

Switch to the SSL tab to add an SSL certificate for the site. Click None under SSL Certificate and select Request A New SSL Certificate (going forward, you can select previously configured certificates from here, too). Provide your email address, agree to the terms and – if required – enable the Force SSL switch to only allow secure (https) connections before clicking Save.

your list a suitably descriptive name and flick the Satisfy Any switch to On. Now switch to the Authorization tab and enter your desired username and password. You can, if you wish, add multiple usernames and passwords, in fact, any of which will allow visitors to access the server.

Click Save, then test your new rule by switching back to the Proxy Hosts tab. Click the vertical ellipsis button next to one of your proxy hosts and choose Edit to make changes. Click under Access List and select your new rule before clicking Save. Now browse to the web address associated with this proxy host and you should see a prompt asking you for a username and password. Enter the details you created, and you should gain access to your server.

Keep it local

Our second example prevents anyone from outside your local network accessing your server. Create a new rule, but this time leave Satisfy Any switched off. This time, switch to the Access tab and type the following into the allow box: **192.168.0.0/24** (substitute **192.168.0** with whatever's used by your network, such as **192.168.1**).

Click Save, add the rule as before to a proxy host, then test: try connecting through your local network using the URL, which should work as normal. Once confirmed, try connecting from outside your home network (by using your mobile's cellular data, say), where you should find access is now denied.

You can combine both examples in a single rule (make sure Satisfy Any is enabled), which would mean anyone on your home network can access the service without limits, while those connecting remotely would be prompted for a username and password before they could gain access.

Thoroughly test that the services continue to work as you'd expect after setting up your rules – some may no longer function correctly, as we discovered while trying to access our Audiobookshelf libraries. The rules may render access from outside your web browser – such as through a mobile app – impossible, too.

Access Lists is a little basic in that it only works with IP addresses and IP address ranges. There's a push among the *Nginx Proxy Manager* community to include GeoIP2, a module that can be used to restrict users by geographical location, enabling you to – for example – block access from outside your home country. Another workaround – if your router supports it (Synology ones do) – can be found in the Quick Tip *(page 57)*.

Beyond proxy hosts

Proxy hosts are specifically designed for web (http/https) traffic that needs directing to a specific device and port. A broader option for non-web traffic can be found using Streams *(outlined in the box below)*.

Nginx Proxy Manager supports two further types of host. Redirection Hosts enable you to redirect traffic from one domain to another (the domain redirected must be configured to point to your home network), while 404 Hosts allow you to redirect domains pointing at your home network to a customised error page.

This error page is defined under Settings > Default Site – by default, you redirect users to a simple (and irrelevant) congratulations page that's a reminder to set up the host for access, but you can replace it with a generic 404 error page, redirect to another site or insert your own message using custom HTML code.

QUICK TIP

Visit https://nginxproxymanager.com for a comprehensive introduction to *Nginx Proxy Manager*. You'll find setup instructions as well as a link to the project's GitHub page, where you'll discover an engaged community and a project frequently updated with new features.

» SET UP STREAMS

If you want to manage non-web traffic, such as a *Syncthing* peer-to-peer connection or a game server, you need to set up Streams. These funnel TCP and/or UDP traffic from a specified port outside your network to a specified device within your LAN.

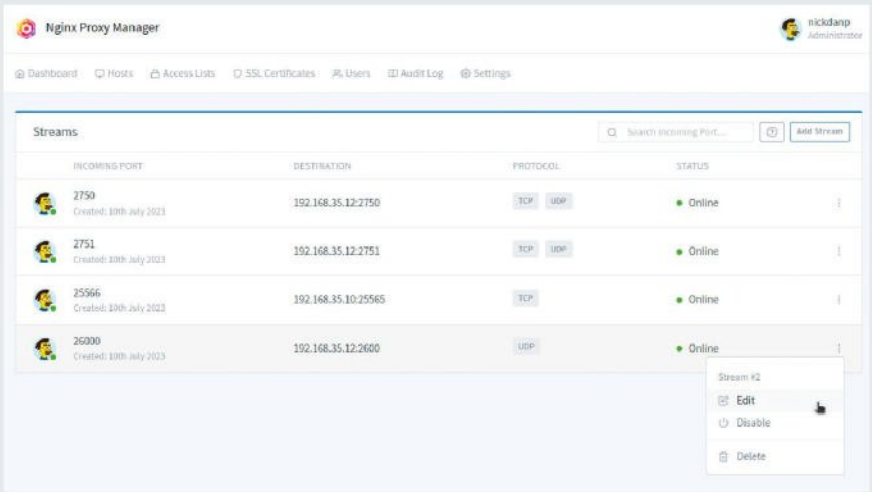
First, you need to configure your router to forward any traffic on those ports to the server running *Nginx Proxy Manager*. Once in place, you need to shut down and destroy the container:

```
$ docker stop nginx-proxy-manager
$ docker remove nginx-proxy-manager
```

Next, edit the *Docker* script *(see first box)* to add the necessary `-p` lines to open the required ports – note the use of `/udp` and `/tcp` to denote the protocols used. This example opens port 1000 for both UDP and TCP traffic:

```
-p 1000:1000/udp \
-p 1000:1000/tcp \
```

Copy and paste the edited script into your terminal window to recreate the container with the port(s) now accessible, then navigate to Hosts > Streams. Click Add Stream to set it up in a similar way to a proxy host, except this time all you need to do is supply the incoming port, forward host and port, plus which protocols to use. Click Save.



You need to configure individual Streams for every single port you want to redirect UDP and TCP traffic on

» **ACCESS US INSIDE YOUR HOME...** Subscribe now at **http://bit.ly/LinuxFormat**

GPRTK.GPIO

Credit: https://github.com/PiSupply/Ryanteck

Add Raspberry Pi GPIO to your PC

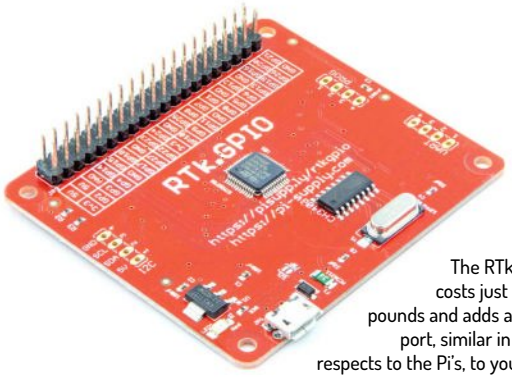
PCs don't offer a GPIO header like the Raspberry Pi, but **Mike Bedford** reveals that a low-cost add-on is all you need to join in the fun.

OUR EXPERT

Mike Bedford uses both PCs and Raspberry Pis, so is quite taken with this method of providing convergence between the two platforms.

The Raspberry Pi might be the poor relation to the PC in performance, but that diminutive single-board computer wins hands down in one area. While a PC's interfacing options are limited to USB, video, audio and wired and/or wireless networking, the Pi adds the all-important GPIO. That, in turn, enables it to be used to control and monitor external electronics and sensors, and thereby work in conjunction with pretty much any equipment.

Thanks to the Ryanteck RTk.GPIO board, all that changes, and an ordinary PC can, like the Pi, be used with external electronics and hardware. Here we're going to see how to use the RTk.GPIO, irrespective of whether or not you're a Raspberry Pi user. If you're a Pi user, this board enables you to migrate some of your projects to your PC, or use some of your Pi HATs – the add-on boards that plug into the Pi's GPIO connector – with a PC. Alternatively, if you're firmly in the PC-only camp, the RTk.GPIO gives you the opportunity to get some experience of real-world interfacing.



The RTk.GPIO costs just a few pounds and adds a GPIO port, similar in most respects to the Pi's, to your PC.

Add-on hardware

While PCs don't have a general purpose input/output (GPIO) port as standard, various hardware solutions are available to add this missing functionality. Traditional products that do this are generally referred to as data acquisition, or DAQ, devices, and connect via USB or occasionally plug into the motherboard. They usually provide digital and analogue I/O so, in that respect, are more powerful than the Pi's GPIO. However, they don't offer any level of compatibility with the Pi GPIO, so migrating projects or HATs isn't easy. However, the biggest drawback is cost. Products start upwards of £100 and rise to several thousands.

Lower performance products aimed at hobby use are available, and we looked at a couple of Adafruit

QUICK TIP

We didn't have a sufficiently demanding application to put it to the test, but we did note some online postings about the RTk.GPIO that compared its response time unfavourably to that of the Pi. However, we suspect it won't be an issue for most users' applications.

» PWM BASICS

The Raspberry Pi's GPIO header is entirely digital, so there are no analogue outputs or inputs. However, using PWM – which the Pi supports, but the RTk.GPIO can only provide via software – it's possible to generate an analogue output signal on a GPIO output pin, albeit with limitations. Given that not all Pi users will be familiar with PWM, here's a brief intro.

PWM stands for pulse width modulation, and the diagram *(right)* shows how it works. In the top part, the output pin is shown as set to a binary 0, so the pulse width is 0% and the voltage is 3.3V. In the middle part, the PWM has a 50% pulse width, so the average output voltage is half of 3.3V, namely 1.65V. The other parts show 25%, 75% and 100% widths. If the frequency of the pulses is high enough, and the requirement isn't too demanding, this works as well as outputting a true analogue voltage. If you use it to drive an LED, for example, and the frequency is at least 100Hz, we wouldn't see it flickering on and off, so it could be used for dimming the LED.

Better approximation to a steady analogue voltage can be achieved using an RC (resistor-capacitor) circuit to average out the pulses and give something more like the dotted red lines. This reduces the speed at which the analogue voltage can change, although using a higher PWM frequency, and thus a lower value of capacitor, helps. This means the RTk.PIO's slower software-based PWM isn't as good as the Pi's hardware PWM.

CREDIT: ELEGOO

products in **LXF274**, namely the FT232H (12 digital I/Os, SPI and I2C, £14.50) and the MCP2221A (four digital I/Os, three analogue inputs and two analogue outputs, although these all share the same four pins, UART and I2C, £6.30). Like the professional DAQ products, though, these are not Pi compatible. However, the product we're featuring here ticks all the boxes – well, most of them. The Ryanteck RTk.GPIO isn't a new product but, we suspect, few PC or Pi users are familiar with it, so we thought it was time to change all that.

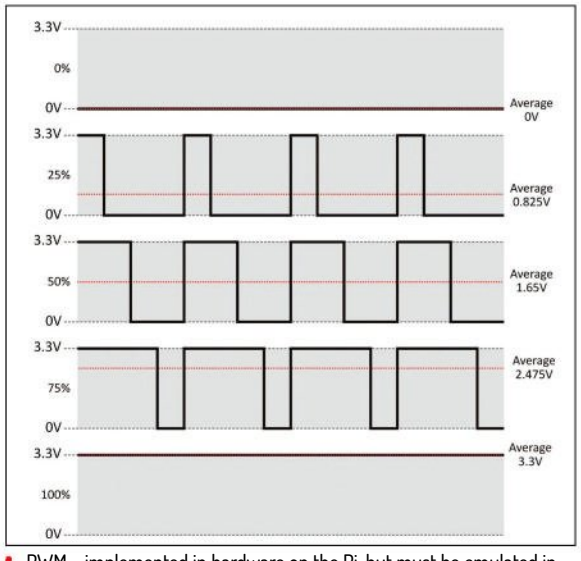
And finally, before homing on our chosen product, we really ought to mention one other interesting option, albeit somewhat different from all the other solutions. This involves using an actual Raspberry Pi's GPIO but controlling it from your PC – we look briefly at this in the Pi GPIO box *(page 63)*.

Ryanteck RTk.GPIO

The Ryanteck RTk.GPIO is available from various sources in several counties – ours came from Pi Supply where it costs £4.99. It's a circuit board a little smaller than a Raspberry Pi, that connects to your PC via USB, which also supplies power to the board. It has a 40-pin male header, exactly the same size as that on a Pi, and at first sight, the pins seem to have the same functions. So far so good for compatibility – taken together these two facts should allow Raspberry Pi projects or HATs to be transferred to a PC. While this is often the case, there are a few differences, however, between the signals on the RTk.GPIO's header's pins and those on a Pi – see the section aimed at Raspberry Pi users for details. If you aren't a Raspberry Pi user and are just looking for a way of learning about physical computing on your PC, these differences needn't concern you unduly. After all, while you'll probably use some published code for inspiration and as a starting point, you'll probably be writing your own code and can tailor this to the capabilities of the RTk.GPIO. However, if you're intending to reuse code you or someone else has written for a Pi, it might require significant modification, depending on which GPIO features you use and, similarly, some HATs won't work.

Installation instructions are provided at **https://bit.ly/lxf306supply** and, in our experience, it's a trouble-free process. There's a sample bit of code at the end of the instructions – it goes by the name of LED Test Example – and we recommend that you try that out, so you have confidence that the hardware is working correctly. Bear in mind, though, that the sample code has some extraneous characters at the end – the `</pre>` on the last line – and you should delete them. Also, if you're a PC user who isn't familiar with breadboarding, be sure to read the next section before trying out the sample code, because it won't necessarily be easy to replicate the breadboard circuit shown in the photo if you don't know much about breadboards, LEDs or resistors.

If you're a PC user with no previous electronics experience, a cheap starter kit, like this one from ELEGOO, is highly recommended.



PWM – implemented in hardware on the Pi, but must be emulated in software on the RTk.GPIO – allows for pseudo-analogue output signals.

PC users

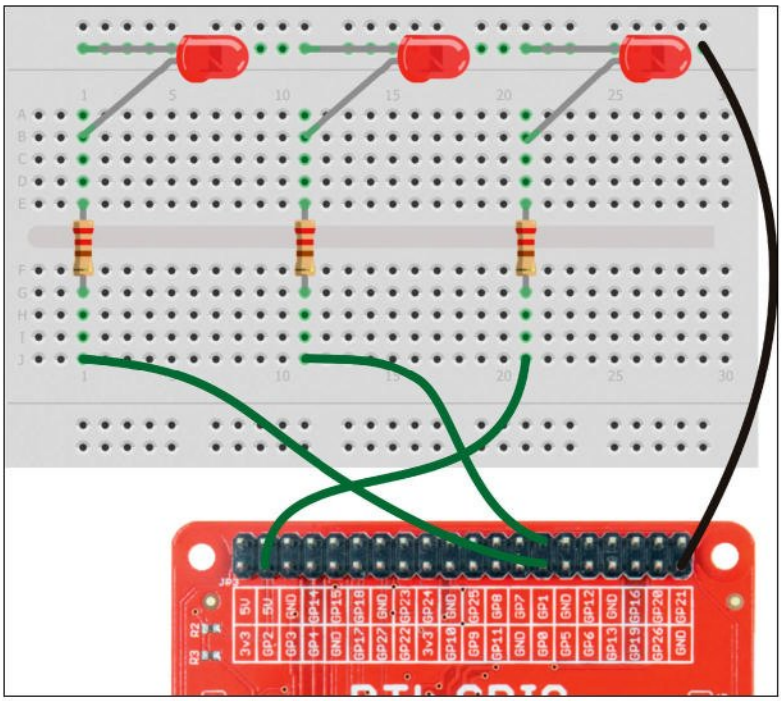
If you're a Pi user, wiring up some electronic circuitry on a breadboard and connecting it to your Pi's GPIO header will be second nature. However, we can't assume all PC users would have this knowledge, and hence would know how to connect external electronics to an RTk.GPIO. So, if you come into that category, this section is for you. As a word of introduction, though, we should warn you that we're only showing you how to build the circuitry necessary to use the sample software that we discussed in the previous section.

A breadboard enables you to connect electronic components together and to a single-board computer, in this case an RTk.GPIO. Sure, you could do that by soldering, and that would be appropriate for something you're going to keep, but breadboarding is a better way for trying things out. For a start, you don't need to buy a soldering iron and learn how to use it. And second, the components are connected by simply plugging their leads into holes on the breadboard, so when you're done, you can unplug them and save them for reuse.

If you've never interfaced to computers before and not dabbled in electronics, you won't have the

QUICK TIP

It doesn't provide a solution for analogue inputs, though, and you might assume that an ADC HAT would do the trick. Beware, though, because although some use the GPIO header's I2C interface so should be OK, others use SPI, which isn't supported on the RTk.GPIO.



After you've installed the RTk.GPIO, be sure to try it out using the LED Test Example code and a breadboard layout like this one.

necessary electronic components at hand. Knowing what you might need at the outset won't be obvious, so we recommend buying a starter pack. One you might like to invest in is the ELEGOO Fun Kit (**www.elegoo.com**), which costs about £13 and should provide all you need for your first steps in prototyping, breadboard included.

The circuit needed to use the LED Test Example is shown, as it might be wired up on a breadboard, in the diagram *(above).* This is much easier to interpret and copy than the photo that accompanies the LED Test Example code online, and we've extended it slightly by using three LEDs instead of one. You can connect them to any three GPIO pins, although if you use three with consecutive numbers, they'll light up immediately after each other as you continue to press the Enter key.

In addition to the breadboard and the RTk.GPIO, you'll notice that there are only six components plus the various leads that connect things together. The interconnecting leads on the breadboard have a pin at both ends, because connections on the breadboard are made by plugging into its holes, but leads with a pin on one end and a socket on the other are used to connect to the RTk.GPIO, because its GPIO header has pins.

The six components are three red LEDs and three resistors. The resistors limit the current that flows through the LEDs, thereby preventing them from burning out. The value of 220Ω, which isn't critical, is suitable for a red LED being driven from a +3.3V signal, as it would be from a Pi or RTk.GPIO. Note that LEDs have a cathode and an anode, and they must be connected the right way round, otherwise they won't light up. The cathode is identified by being adjacent to the flat bit on its round cylindrical body, and also by virtue of it having the shorter of the two leads.

These internal connections are shown in green in the *Fritzing* diagram, but in the real world you just have to visualise them. However, this is easy with a simple circuit, if you remember the rules. Holes on the top two and bottom two horizontal rows are connected together internally, and are intended for power connections – 0V (GND) and either +3.3V or +5V, although you'll mostly be using +3.3V with the Pi or RTk.GPIO. Similarly, holes in the same vertical columns are connected together internally, although those internal connections have a break at the centre of the board – the gutter – in which there are no holes.

QUICK TIP

If you prefer Arduino-style physical computing on your PC, instead of Pi-style, you might be interested in *Virtual-GPIO* – see https://github.com/A33J/virtual-GPIO. In fact, it actually is an Arduino, running a sketch to take commands from the PC and implement them locally on its GPIO.

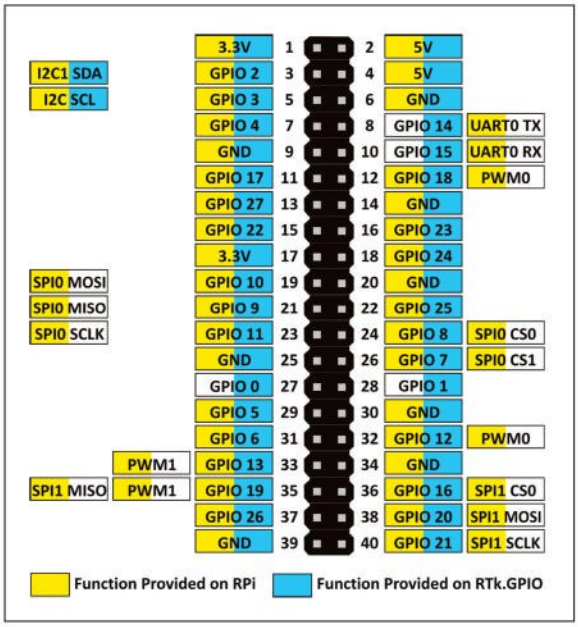
Raspberry Pi users

The first thing we need to explain to Pi users is how the RTk.GPIO's GPIO header differs from that of the Pi. It might seem a bit negative to explain which of the Pi's GPIO-related functions the RTk.GPIO doesn't support, but if that prevents you buying something that won't meet your needs, it's justified, so here's the low-down in brief. Note that our reference to the Pi's 40-pin header refers specifically to the Pi B+ and the more recent versions, not the 26-pin original Pi B+.

Commonly, the term GPIO is used to refer to a signal pin that can be configured as a digital input or output. Things aren't quite that simple, though, because the Pi's header provides access to more than just GPIO signals. Several pins provide access to power supplies for use by external circuitry, specifically +5V, +3.3V and GND (0V). Some of the GPIO signals can be used to output PWM signals. Two of the pins are used to access a UART, which provides serial signals. And finally, some of the pins on the header can be configured for use as various alternative functions, specifically the I2C and SPI serial interfaces.

The RTk.GPIO's GPIO, by which we mean its 40-pin header that provides access to various signals, isn't exactly the same as that of the Pi. Certainly, it carries all the same GPIO signals on the same pins, it carries the same power supplies on the same pins, and it provides access to I2C on the same pins. However, the UART and SPI interfaces are drop-offs; four additional GPIOs are provided on the pins that carry the Pi's UART signals and a couple that are reserved.

Despite there being no UART on the 40-pin header, there is an entirely separate four-pin header labelled


3.3V 1 2 5V
I2C1 SDA GPIO 2 3 4 5V
I2C SCL GPIO 3 5 6 GND
GPIO 4 7 8 GPIO 14 UART0 TX
GND 9 10 GPIO 15 UART0 RX
GPIO 17 11 12 GPIO 18 PWM0
GPIO 27 13 14 GND
GPIO 22 15 16 GPIO 23
3.3V 17 18 GPIO 24
SPI0 MOSI GPIO 10 19 20 GND
SPI0 MISO GPIO 9 21 22 GPIO 25
SPI0 SCLK GPIO 11 23 24 GPIO 8 SPI0 CS0
GND 25 26 GPIO 7 SPI0 CS1
GPIO 0 27 28 GPIO 1
GPIO 5 29 30 GND
GPIO 6 31 32 GPIO 12 PWM0
PWM1 GPIO 13 33 34 GND
SPI1 MISO PWM1 GPIO 19 35 36 GPIO 16 SPI1 CS0
GPIO 26 37 38 GPIO 20 SPI1 MOSI
GND 39 40 GPIO 21 SPI1 SCLK
Function Provided on RPi
Function Provided on RTk.GPIO

Although the basic digital I/O pins are the same, some of the Pi's alternative GPIO functions aren't replicated on the RTk.GPIO.

» USE AN ACTUAL PI'S GPIO

OK, we admit that it doesn't provide a solution for the PC user who is determined never to use a Raspberry Pi, but an alternative way to provide Pi-compatible GPIO capability on a PC is to use an actual Raspberry Pi, suitably interfaced to the PC. What's more, unlike the RTk.GPIO board, this method replicates the Pi's 40-pin GPIO header in its entirety, UART, SPI, PWM and all. It would be considerably more expensive than using the RTk.GPIO if you were to use a Raspberry Pi 3 or 4, though. However, a Pi Zero costs around £15 – although you'd pay slightly more for a Zero with pre-soldered GPIO header – a price premium over the RTk.GPIO but one you might consider acceptable, given the 100% compatibility.

The secret is to use a facility on the Pi called Remote GPIO, which hands over control of the GPIO header to an external device. This could be another Pi, but we're more interested in it being a PC running some flavour of Linux. As this is just an aside to the main theme of the article, we're not going to explain how to do that, although you'll find plenty of material online. An article, written by Raspberry Pi's former community manager, Ben Nuttall, and first appearing in *MagPi,* can be found at **https://bennuttall.com/wp-content/uploads/2017/07/remote-gpio.pdf**, and is a good first port of call.

UART, but we understand that the functionality hasn't been implemented in the firmware. Another difference is that none of the GPIO signals can be used to output a PWM signal. However, PWM can be provided in software, but this isn't as good as the Pi's hardware PWM, as explained in the PWM box. This requires a change to Pi-compatible software, but there is some sample software PWM code in the Ryanteck GitHub resource at **https://github.com/PiSupply/Ryanteck/**.

A picture is worth a thousand words, so you may like to peruse the comparison illustration *(below-left),* which makes it easier to see the differences between the signals on the Raspberry Pi's 40-pin header and those on the RTk.GPIO's.

Finally, on the subject of the hardware differences between the Pi's GPIO header and that of the RTk.GPIO, we come to the current sourcing and sinking capability of the GPIO signals, which depends on the on-board processor, which is different for the two products. For common requirements, such as lighting an LED, this need not concern you, but if you have a more sophisticated requirement, you probably ought to delve into this in more detail. Looking at the datasheets for the respective processors and other documentation is really the only way to be sure, but here's the abbreviated version. For the Raspberry Pi, one of the most authoritative sources of information appears to be that at **https://bit.ly/lxf306gpio**, which has aimed to piece together information from several sources, to overcome the perceived deficiencies in the datasheet for the Broadcom SoCs, which are used on board Raspberry Pis.

The bottom line is that, by default, each GPIO pin can sink or source 8mA, although this can be reconfigured to anywhere between 2mA and 16mA. This is subject to a maximum for all the GPIO pins, which, in this case, is 50mA. Turning to the RTk.GPIO, the datasheet for the STM32F030XX family chips, as used on that board, states the following: "The GPIOs (general purpose input/outputs) can sink or source up to +/-8mA, and sink or source up to +/- 20mA (with a relaxed VOL/VOH)." It also warns that these figures are subject to a maximum current for the chip as a whole, which is 120mA. At first sight, this top-level comparison rather suggests that migrating from the Pi to the RTk.GPIO shouldn't pose a problem, because its processor can handle higher currents than the Pi's. However, the fact that the Pi's GPIO pins' current handing capabilities are programmable, and the RTk.GPIO's pins can only handle more than 8mA with "a relaxed VOL/VOH", suggests that more scrutiny might be required.

UNICORN HAT
An equine banquet of 64 WS2812B RGB LEDs in a handy, expressive 8x8 grid
Made in Sheffield, UK
http://pimoroni.com
PIMORONI

Many HATs designed for Pis can be used with the RTk.GPIO, but not all, so be sure to do your homework first.

Turning to the software, if you've perused the LED Test Example sample code provided for the RTk.GPIO, which we suggested you try out to make sure your installation worked correctly, you'll probably have noticed something you might not have expected. In particular, where you were expecting to see `import RPi.GPIO as GPIO`, you'll have seen `import RTk.GPIO as GPIO`. However, the RTk.GPIO library aims to replicate most functions found in the RPi.GPIO library, so simply altering the code to import the different RTk.GPIO library should be all that's necessary to get your Raspberry Pi code working on the RTk.GPIO.

We should also say a few words about migrating HATs from the Raspberry Pi to the RTk.GPIO board. And the bottom line is that you should study the HAT manufacturer's data and/or schematic to check whether the HAT uses any of the functions on the Pi's GPIO header that are not replicated on the Ryanteck product. So, for example, if the HAT makes use of the Pi's UART or SPI, it's not going to work with the RTk.GPIO. It's possible, however, that sufficient information might not be provided with all HATs to make this judgement, and you might not have sufficient experience in electronics to figure it out yourself from the schematic, if such a diagram is made available, that is. In such a case, we'd be inclined to suggest that if you're interested in using a HAT you already own, you simply try it out, subject to swapping the RPi.GPIO library to the RTk.GPIO equivalent. On the other hand, if you're considering buying a HAT specifically for use with the RTk.GPIO board, and adequate technical information isn't available, if it costs more than the odd pound or so, you might feel it wouldn't be wise to buy one speculatively. LXF

CREDIT: Gareth Halfacree/https://commons.wikimedia.org

» **IMPROVE YOUR LINUX SKILLS** Subscribe now at **http://bit.ly/LinuxFormat**

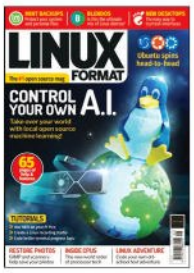
BACK ISSUES » MISSED ONE?

ISSUE 305
Summer 2023

Product code:
LXFDB0305

In the magazine
Don't let Big Tech be in control of your documents – find out how to run your own office apps in the cloud. Plus, learn how Linux certification can kick-start your career, step back in time with our look at the evolution of coding, and read our recommendations for the best open source CAD packages, not forgetting our usual packed tutorial, review and Raspberry Pi sections.

ISSUE 304
August 2023



Product code:
LXFDB0304

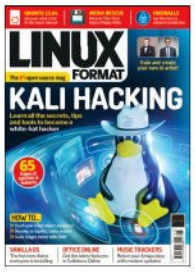
In the magazine
Run your own AI and take over your world with local open source machine learning, discover the diverse delights of BlendOS, and learn all about AMD, Intel and Apple's different takes on CPU design. Plus, we have an in-depth interview with Collabora's Michael Meeks, a *Roundup* of official Ubuntu spins, and tutorials on everything from restoring old photos to coding a text adventure.

ISSUE 303
July 2023

Product code:
LXFDB0303

In the magazine
Trim down your Ubuntu installation for a stripped-back, super-slick and ultra-fast Linux experience. Plus, discover which beginner programming language is the best, read all about OpenSUSE's ALP distro, and find out why *Podman* might be better than *Docker.* And don't forget to try our tutorials on terminal-based to-do lists, searching file contents, making mazes in C, and loads more.

ISSUE 302
June 2023



Product code:
LXFDB0302

In the magazine
Master hacking with Kali Linux, discover the artful ways of *Stable Diffusion,* try a taste of the anything-but-plain Vanilla OS and find out what's new in the online office world. Plus, pick up a whole bunch of new skills thanks to tutorials on speeding up text-based tasks, rescuing data from floppies, repairing images in *GIMP* and much more, and learn which is the hottest firewall distro in our *Roundup.*

ISSUE 301
May 2023

Product code:
LXFDB0301

In the magazine
Save your old PC by installing Linux Mint on ageing hardware. Also in this issue, we explore the programming language leaderboard, look at keeping fit the open source way, feature a *Roundup* of lightweight desktops and spoil you with tutorials on everything from image processing and music streaming to programming a 3D chess game and controlling your DIY robot.

ISSUE 300
April 2023

Product code:
LXFDB0300

In the magazine
The 300th issue celebrates the very best open source apps and revisits the birth of the free software movement. We also take an in-depth look at the ups and downs of *Firefox* and enjoy some retro distro reviews from the early days of the magazine. There's a *Roundup* of current key distros, too, plus tutorials on Debian Hurd, *GIMP,* Mastodon and coding your own *Pong* game, to name just a few.

To order, visit www.magazinesdirect.com

Select **Single Issues** from the tab menu, then select **Linux Format.**

Or call the back issues hotline on **0330 333 1113**
or **+44 (0)330 333 1113** for overseas orders.

Quote the product code shown above and have your credit or debit card details ready.

USA? EU? THE MOON*?

UK readers turn to **p18**

SUBSCRIBE!

Don't wait for the latest issue to reach your local store – subscribe today and let *Linux Format* fly straight to you. Faster, cheaper and with DRM-free archive access!

3D MODELLING
We test the best CAD apps for 3D printers
SAFE NETWORK
Get up and running with a home firewall
RUNNING RISC-V
The low-cost way to run the hottest CPU
LINUX FORMAT
The #1 open source mag
Become a Linux professional
ESCAPE GOOGL
Free yourself from Microsoft and Google
Run your own office apps in the cloud
Take back data and privacy control!
100 pages of Linux tricks, tips & more!
TUTORIALS
Get started with Joplin notes
Create funky plots with Pascal
Extend your own text adventure game
CODING HISTORY
Explore the evolution of programming code
INSIDE DEBIAN 12
Discover what's hot in the latest release
GITHUB WEBSITES
How to create a static site for your projects

3 GREAT WAYS TO SUBSCRIBE

Print, digital-only, and print+digital bundles!

» USA
From $132
For 13 issues

» REST OF THE WORLD
From $132
For 13 issues

» EUROPE
From €100
For 13 issues

IT'S EASY TO SUBSCRIBE!

Visit **www.magazinesdirect.com/linux-format**

Call **+44 0330 333 1113**

Lines open Monday-Friday, 9am-5pm, UK time.

*We don't actually deliver to the Moon. Yet.

BLENDER

Credit: www.blender.org

Render real-world 3D maps in Blender

Intrepid **Michael Reed** explores real-world maps by never leaving his home and rendering them in Blender using various data sets.

OUR EXPERT

Michael Reed has spent more time working on his *Blender* skills than an award-winning chef.

3D graphics editor *Blender* can adapt to many different tasks, so we're going to use it to render 3D maps sourced from real-world data. The process is fairly simple and doesn't require much in the way of *Blender* experience. You can take the map and height data resources from a variety of sources. What you end up with is a 3D terrain you can zoom, rotate, fly through and render. With *Blender's* facilities, you can even animate the resulting object.

We need a 2D map image of some kind, along with a heightmap. In this case, a heightmap is a bitmap image in which the brightness of each pixel represents the height of that part of the map. When we combine the two, we have a 3D model of a map, ready for rendering.

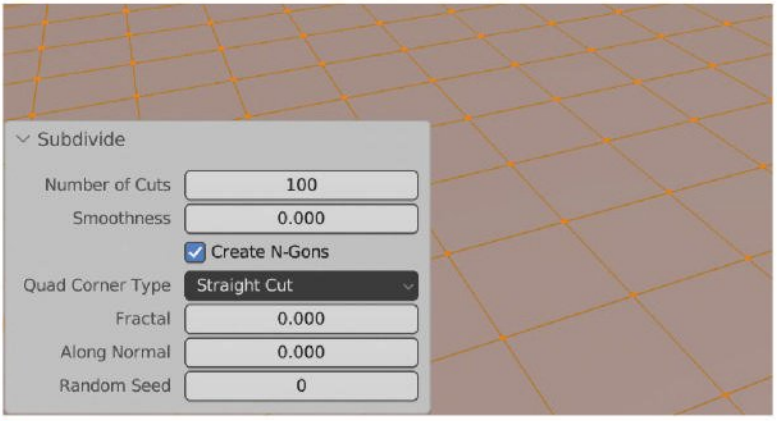
You can use a variety of free online data sources for the map image and the height data. You're not even confined to using it for real-world map rendering because you can also use a variety of computer-generated, mathematical images as input sources.

QUICK TIP

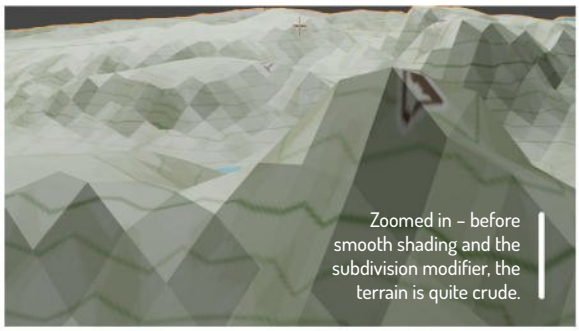
Our *Blender* doesn't look quite the same as your *Blender?* We selected the light theme so that it looks better in the magazine.

Acquiring the data

The first thing we need to do is to acquire some map data. For the method that we're going to use, we require two images: a heightmap and a second image to 'drape' over that heightmap. The great thing about working like this is that it gives us a lot of flexibility in terms of the images that we can use. See the box *(opposite)* for tips on sourcing suitable heightmap and map image files. One thing to look out for: make sure you are selecting a geographical region that is mountainous so that you can see the 3D effect.



Once subdivided, the surface of this plane is made up of many smaller rectangular faces.



Zoomed in – before smooth shading and the subdivision modifier, the terrain is quite crude.

A quick cleanup

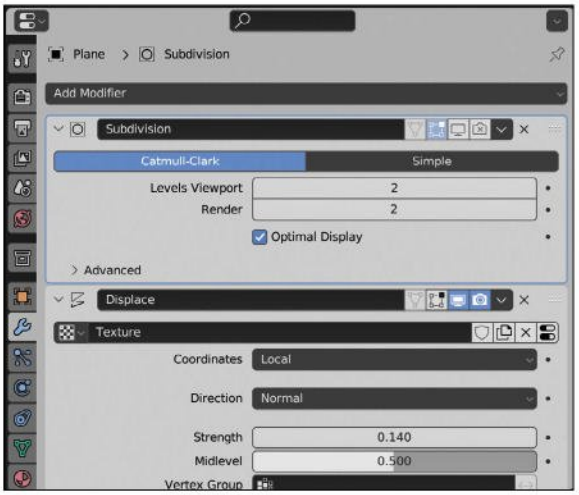
It's worth giving the heightmap and the map image data a quick clean up, and for this, we'll use the image editor *GIMP.* The source images we're dealing with might have something like stray text or an unwanted detail here and there. Most of these instructions can be applied equally to both the map image and heightmap.

Any elements we need to remove can be worked on with *GIMP's* Paintbrush tool. Use the Eyedropper to select the background colour of the map and then remove anything we don't need with the Paintbrush. The Clone tool and the Healing tool may also be useful for this work. The only tools we shouldn't use in *GIMP* are those that change the size of the image, such as the crop or scale tools. That's because we need the heightmap and the map image to line up perfectly.

When editing the heightmap in *GIMP,* changing the contrast changes the dynamic between the highest part of the terrain and the lowest. There are some changes to this relationship that can be made from within *Blender,* but you can get into more detail in *GIMP.* Increase the contrast slider (in Colours > Brightness-Contrast) to make the dark parts darker (lower) and the light parts brighter (higher).

Creating the plane

Load up *Blender*, and on the splash dialog that pops up, choose General as the document type. Our first task is now to remove the default *Blender* cube, which you should see in the viewport (which is what we call the



The right-hand tabbed sidebar is dominated by the Properties editor, which is where you'll be spending a lot of time.

A rendering of Ambleside from data from https://heightmap.skydark.pl. We added some extra contrast to the final render in GIMP.

main 3D window). This gives us an empty scene to work with. To delete the cube, click on it in the viewport and press Delete.

Add a plane to the scene (Add > Mesh > Plane in the menu at the top of the viewport). A plane is a completely flat object that we shall use for our 3D map. At the moment, though, it's too basic for our eventual purpose. It has four edges and four corners. These corners are specified by X, Y, Z points called vertices. Obviously, we can't have a 3D map on a completely flat plane, and the only part of the plane we can vary at the moment is the position of those four vertices.

To get around the problem of the plane being completely flat, we'll subdivide it. In other words, we want the plane to be made up of thousands of little squares, like a piece of graph paper. Each corner of those squares can be set to any height we need, giving a 3D terrain.

Make sure that the plane is selected (left-click it to be sure) and move from Object mode into Edit mode (press Tab). Right-click on the plane and select Subdivide. Initially, this only subdivides once, splitting the plane into four parts.

When you clicked on Subdivide, you might have noticed a small, collapsed dialog box appear in the bottom-left. Press the arrow on the side of it to expand it. From here, we can be more specific about how we want the subdivide function to work.

The only parameter we need to change in this case is the number of subdivisions. Put 100 into the parameter box (double-click to edit it) and press Return. You should now see that our plane is subdivided into thousands of squares. Press Tab again to move back into Object mode. A note here: most of this tutorial is in Object mode, so make sure you're in the right mode at all times; the current mode is indicated in the top-left corner of the viewport.

A matter of import

We'll add the map image before working with the heightmap. The map image is a texture, but *Blender* doesn't allow us to add textures directly to an object such as the plane. Instead, we'll create a material, add the texture to the material, then attach the material to the object. This might sound like *Blender* is adding an extra stage, but it's worth it because it means we can tweak the overall appearance of surfaces, including the attached texture, and can have more than one texture. See the step-by-step *(over the page)* for how to attach the map image to the plane's surface.

Make sure that the plane that you are using for your map has the same aspect ratio as your bitmaps. On the

QUICK TIP

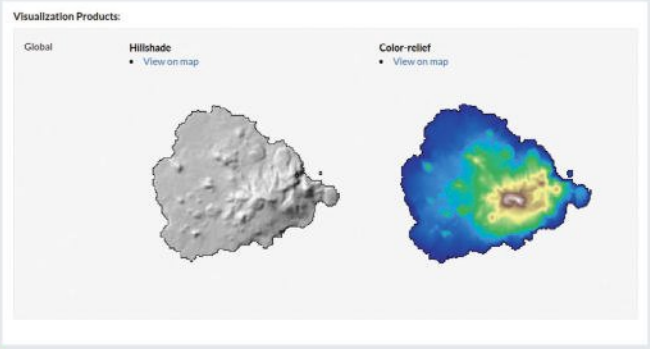
You only need a simple set of controls for this project. Move the mouse with the middle button pressed to rotate the view, and press the middle mouse button and Shift to move around.

» ACQUIRING THE DATA

Provided the resolution is reasonable, you can use any source as long as you end up with a map image that matches up with a heightmap. This means you can acquire suitable bitmaps by simply capturing the entire screen, if nothing else. There may be some legal restrictions affecting images obtained in this manner, so check the source website for information on this.

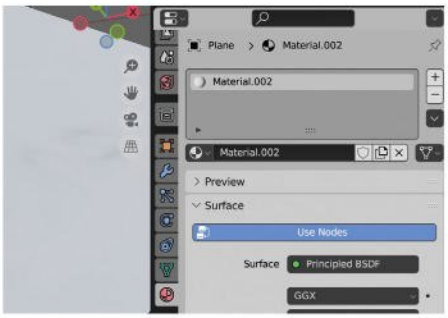
The website at **https://heightmap.skydark.pl** was designed to provide map data for the city-building game *Cities: Skylines*. Download the PNG heightmap and the map image.

OpenTopography (**https://opentopography.org**) is tricky to navigate. Some of the best data is located in the Shuttle Radar Topography Mission (SRTM GL3) archive found by listing all datasets on the Data page. Select IMG as the output and Hillshade and Color-Relief. Right-click and save the images.



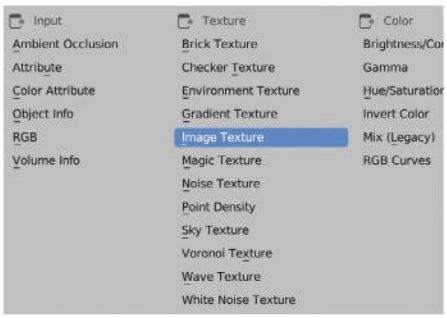
Getting ready to right-click and save from the OpenTopography site. Don't worry about the images being low resolution as they are simply source data.

IMPORT AN IMAGE



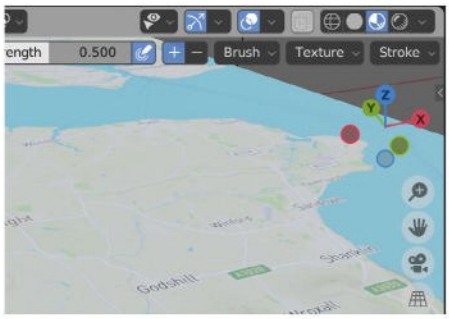
1 Create the material

The sidebar on the right-hand side of the viewport is where we alter the properties of objects in a *Blender* scene, and it's divided into tabs that are represented by icons. To find out which icon is which, hover over it with the mouse. Select the Material tab. Press the + New button to create a new material.



2 Add the texture

This material has a flat grey colour assigned to it, but we need to assign a texture (our map) to it instead. Click on the yellow dot within the Base Color property of the Material tab and select Image Texture. There should be some options for adding a texture. Click on the Open button and select the map image texture.



3 Switch to Material Preview

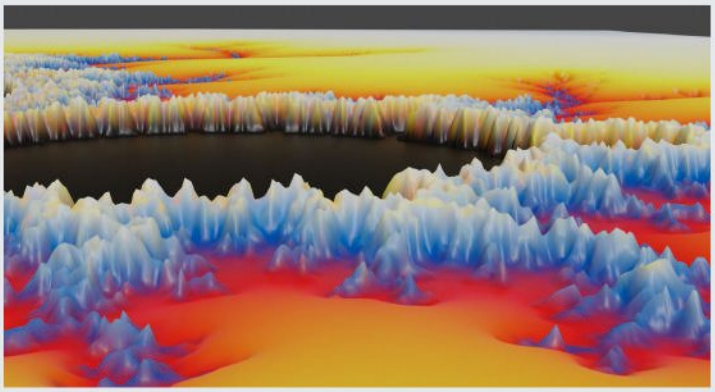
You won't see a change at first, because the viewport defaults to being in Solid mode by default, and it doesn't show any materials. Switch the viewport to Material Preview mode on the small icon bar in the top-right corner of the viewport. You should now see the map texture on the surface of the plane.

» MANDELBROT MAPS

The process that we've employed (subdividing a surface, applying a texture to it, and applying displacement to the surface) can be used on source images other than maps. For example, many fractal-rendering applications can output a monochrome version of the same image. Failing that, it's often possible to use *GIMP* to reduce the fractal image to monochrome (Colours > Desaturate > Desaturate). When we tried this we found that the brightness was sometimes the wrong way around, with the lowest parts of the map brighter than the highest parts. It was simple to invert this with *GIMP* (Colors > Invert).

The only problem was that this tended to make the base of the map white rather than black, which would place it at the maximum height rather than the minimum. The solution was to click on one of these areas with the Select By Color tool selected. This selects all of the white regions in the image. Reapply the invert function to change these areas from white to black.

Some Mandelbrot renderers produced output that was more suitable than others. *XaoS* produced output that was too 'bitty' and detailed for what we wanted, while *QFractalNow* produced a more suitable output.



We rendered this fractal in QFractalNow (https://fractalnow.sourceforge.io) and made a heightmap in GIMP.

right side panel in *Blender*, use the Object tab. For example, for a typical 1080p image, change the scale to X: 1.6 and Y: 0.9. This gives the plane a 16:9 ratio.

Once you have the map image texture on the plane, it's time to add the heightmap and make it affect the heights of the vertices on the plane's surface. To do this we use displacement, where one input (the heightmap) affects the shape of an object (the plane).

Select the Modifier tab in the Properties sidebar. A modifier alters the way something is rendered in the viewport and in a final render without altering the underlying object. Select Add Modifier and choose Displace. Don't be concerned if this causes the plane to move upwards within the viewport.

In the sidebar there should now be a set of properties to edit for the Displace modifier, and the procedure of adding the heightmap as a texture is like that for adding the map image texture. In this case, click on + New to add a new texture. Switch to the Texture tab, click on Open and add the heightmap texture. You should see that the height of the plane's surface undulates according to the relative brightness of different regions of the heightmap. You can alter the amount of effect that the heightmap applies to the plane's surface by altering the Strength and Midlevel parameters of the Displace modifier dialog. Most of the *Blender* parameter sliders are very sensitive, so double-click and then manually enter the parameter values or shift-drag the sliders for fine control.

Improving detail

If you zoom into the surface of the plane, you might notice height variations are rather crudely represented and polygonal in nature. That's hardly surprising, as the vertices of the plane's surface only amount to a 100x 100 resolution. The first thing to do to improve the look of the 3D map is to select smooth shading by right-clicking on the plane and selecting Shade Smooth.

We can also increase the number of vertices that make up the surface of the plane by using another

modifier. As before, add the modifier from the Modifier tab, but this time, select Subdivision Surface. Modifiers are processed in order, so drag the Subdivision Surface modifier so that it sits above the Displace modifier. What the Subdivision Surface modifier does is subdivide the surface again, but like all modifiers, it does not alter the underlying object; these are simply changes that are made when the scene is shown in the viewport or rendered to a file. Notice that this modifier has two parameters, one for how much subdivision to carry out in the viewport and one for the final render. In both cases, changing this to 2 is usually sufficient.

Consider going back into the Material editor and tweaking some of the parameters there. As you reduce the Roughness parameter, the material becomes shinier and eventually looks like glass. As you increase it, the material becomes more matte.

Lights, camera, action

Now that we have the 3D map on the surface of a plane, we need to get it into a state where it can be rendered. The first issue to consider is the lighting. *Blender* defaults to using a single point light because this gives even light to the scene. An object like our 3D map would benefit from a light that provides more contrast and introduces shadows into the scene. We need a light that is small and close to the subject. To see the effect of the lighting, make sure the viewport is in Render Preview mode by selecting it in the small icon bar in the top-right area of the screen.

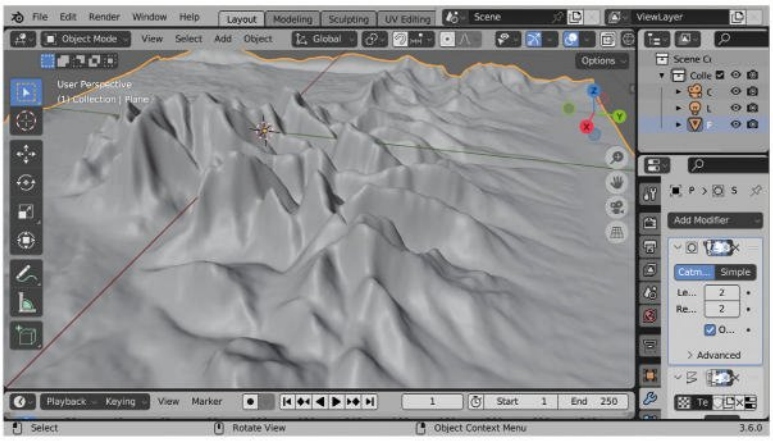
Add an area light to the scene (Add > Light > Area in the menu above the viewport). Immediately, this light should add more contrast and extra shadows. Click on a light to select it either in the viewport or in the Scene Collection panel at the top of the right-hand sidebar, and move a light by pressing G to go into Grab mode.

While you are in Grab mode, you can restrict the movement to a particular axis by pressing X, Y, or Z. It is often convenient to move in one axis, such as Z, which is usually the height, press the left mouse button to fix the chosen position, and then go into Grab mode again and move the object in another axis. These workarounds are needed when moving an object around a 3D space, which is usually skewed at an angle, with a 2D pointing device such as a mouse.

You'll notice that moving the default point light doesn't have much of an effect on the look of the scene, but it looks massively different according to the position of an area light. If the bland-looking point light is overwhelming the effect of the area light, you can turn down its brightness. To do so, click on it to select it and then go to the Data tab of the Properties panel. Here, you can drag it down to a much lower value. If you want a lot of contrast and shadows, you can simply delete the point light entirely. If there are areas of the map that now don't have enough illumination, you can add a second area light and position it appropriately.

Camera placement and rendering

Hopefully, the 3D map looks good at this point. If we want to render the scene, we need to make sure that the camera is pointing in the right place. To see what the camera is looking at now, press 0 on the numeric keypad. This is unlikely to be the optimal viewpoint. The simplest way to move and orientate the camera is to make it so that it reflects the current viewport view. To do this, expand the little View dialog by pressing N. Select the View tab on this dialog, and in here, click Camera To View. You can now move the camera using the same mouse movements that you would normally use to move around a *Blender* scene. Once you have got the perfect angle worked out, don't forget to deselect the Camera To View option.



Viewing the effect of a heightmap displacing the surface of the plane without a material and a texture.

Press F12 to render the scene. Hopefully, the lighting and camera options are to your liking. Finally, save your image (Image > Save As).

Animate it

Blender makes it easy to animate a scene by, for example, rotating an object. The animation timeline is at the bottom, beneath the viewport. Hover over its top edge and drag it upwards to enlarge it.

We can control the rotation of the plane object by changing the Z rotation parameter in the Object tab of the Properties panel. By default, this parameter should be set to zero. If you click the small grey dot at the end of this parameter, a keyframe is added to the timeline.

Edit the Z axis rotation property for the plane object, and change the value from 0 to 360. Now, within the timeline window, drag the cursor to frame 250, the final frame.

Once again, click on the symbol next to the rotation property, which should have changed into a diamond shape. This inserts a keyframe specifying the rotation parameter of 360 on to frame 250 on the timeline.

You can now drag the timeline cursor back to the beginning. When you press the play icon at the top of the timeline window, the plane object rotates 360 degrees. Many object properties can be animated in this way. For example, you could change the position of the camera and fly into the map.

To tell *Blender* that we want to render this animation as a video file, change the output format from PNG to a video format (Sidebar > Output tab). To render the animation, press Ctrl+F12.

Hopefully, we've given you some useful tips for rendering all sorts of map and map-type data in *Blender*. Happy rendering! LXF

QUICK TIP

For this project, we need to keep the object that we're working with (the plane) selected by left-clicking on it. Make sure you're in the right mode (Edit/Object) as well.

» **RENDER THE SHOPS USELESS...** Subscribe now at **http://bit.ly/LinuxFormat**

Stuart Burns is a Linux administrator for a Fortune 500 company, specialising in Linux.

» A ROCKY ROAD AHEAD?

I was a big fan of Red Hat. Until the announcement that it has closed the previously public access to the source code from which RHEL is built in favour of CentOS Stream. The only people allowed to obtain or use the exact version of code are Red Hat customers. See **https://red.ht/3JnJ1nG** for details.

Red Hat is trying to dress this up as a good thing because in place of that it is telling people (including anyone who uses the source to build a 1:1 binary compatible kernel) to use the Red Hat CentOS Stream release code. One of the main reasons Rocky Linux came into existence is the changing of CentOS from a stable identical but free as in beer RHEL release into what is essentially a rolling testing/beta release, making it unsuitable for those using CentOS as they can't afford RHEL licence fees.

This will make it very difficult for Rocky and, perhaps more interestingly, Oracle. Oracle put big money into trying to get large corporations moved to OEL.

Having been involved in a migration to Oracle Red Hat-compatible kernel offerings at cheaper prices than Red Hat, I understand the reasoning behind why Red Hat shouldn't help Oracle take business off Red Hat using Red Hat's own code.

But that understanding is tempered when you remember that Red Hat's ultimate owner is now IBM, a company seen by most as out of touch with the next generation of system designers and administrators. I suspect this will cause more headaches and bad will for Red Hat than additional revenue.

Docker is dead, long live Docker

Docker has been the king of containers but could Podman put an end to Docker's reign?

Everyone knows about *Docker.* In fact, a lot of users conflate containers and images with *Docker*. However, in today's world, there are many alternative choices, including *Podman* from Red Hat. Part of the reason Red Hat created *Podman* was to allow it to run rootless and daemon-less, thereby running more securely by default (a *Docker* group has to have root privileges to run).

Just because it is Red Hat doesn't mean it leaves other Linux vendors out in the cold. For example, performing `sudo apt-search podman` on an Ubuntu server reveals all the default tools to install *Podman.* The building of new images and so on requires more RH tools (all free) but it also includes a cute GUI for container and image management.

The containers work the same thanks to the Open Container Initiative, which details the interoperability and use requirements. It has a *Docker*-compatible command line for those just checking it out, so there should be no issues. It also has a free-to-use GUI that is feature equivalent (in our point of view) to Docker Desktop.

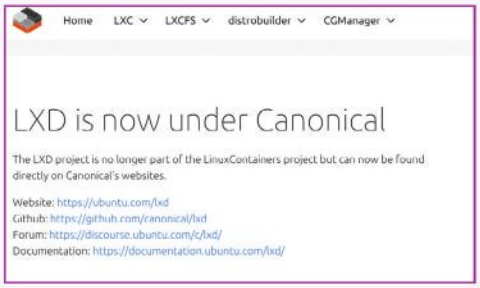
Podman is fully open source and, unlike *Docker*, is free from constraints around licencing. While compatible with *Docker,* it comes with its own build process and tools if someone seeks to build their containers.

A brave new competitor to Docker or just a flash in the pan of containerisation?

LXD gobbled up by Canonical

LXD ownership is transferred to Canonical in the hope of making great strides.

Canonical has taken stewardship of LXD, the extension for managing LXC (Linux Containers). The Linux Containers team explicitly pointed out that everything else LXC will remain under its stewardship: "Canonical, the creator and main contributor of the LXD project, has decided that after over eight years as part of the Linux Containers community, the project would now be better served directly under Canonical's own set of projects."



LXD is now under Canonical's direction. New features inbound or same old LXD?

As to the reason why, it does seem to boil down to the fact that over the last several years, there has been limited development of LXD, and that under Canonical's guidance, it can more tightly integrate it into the various versions of Ubuntu, and extend the functionality to enhance the whole container experience on Linux.

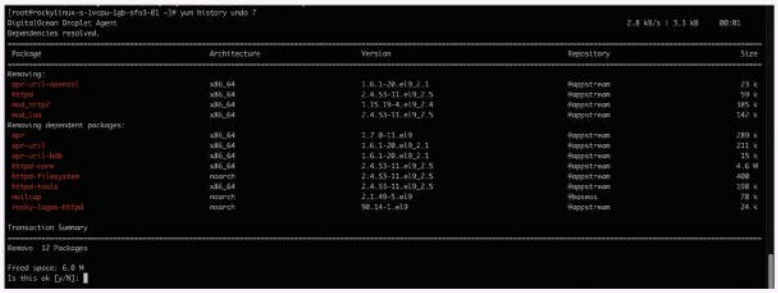
For those using LXD, there is no need to fret. It is still retaining its open source and free beer nature. The Git repository has been moved under Canonical's stewardship. Further details can be found at the LXD site: **https://linuxcontainers.org/lxd/**.

Who installed that?

Yum is more than just a way to install applications – it can do all sorts of interesting stuff.

A lot of Red Hat users never get beyond `yum install tmux -y` or basic package installation. However, when things go wrong, there is a golden *Yum* parameter switch that may help save the day. And *Yum* is also useful for seeing who did what and when!

The *Yum* command has a **history** parameter that is useful for troubleshooting. On an RHEL/Rocky/RH server, it enables administrators to see who ran what command, the issues, the operation with packages and even what the system did during the configuration. Was a package installed, removed, upgraded or removed? *Yum* history can tell you.



Yum undo – a really useful but less well known switch to roll back package installs.

Unfortunately, `yum history` or `dnf history` functionality is unique to *Yum*-based servers. There is no similar functionality in *Apt* package management systems. *Apt* has some commands such as `apt list` but nothing quite as good. A lot of the history of what was installed and when is available in **/var/log/apt/history.log** if needed, but be aware that this log may get rotated, so long-term install information may be lost.

As a real-world example, a few weeks ago we had a scenario where a package on a system was being upgraded but not fully completed, but we didn't know what was causing the issue. Using the `yum history` command, we were able to locate what was going on and could address the issue. It turned out that it was erasing a package but not completing the replacement of said package.

By itself, the `yum history` command shows what has been happening and the number of changes *Yum* implemented. A fresh VM with just an *Apache2* installation is shown below. Using `sudo yum history` gives a useful tabular output. Sometimes the output truncates to a single letter or series of letters depending on what the *Yum* command did: I is install, U is upgraded, E is erased, R is reinstalled and D is downgraded.

Each ID represents the output of some *Yum* command interaction. The ID field is important as many of the commands need a transaction ID to work against.

If you want to dive in deeper than just the command line and date and time, there is a new parameter to use: **info**. To see the changes made in the initial system update, for example, we can use `yum history info 2` to give us all the details of packages that were upgraded, installed or removed, as well as the completion status.

Notice the User field – on a properly configured RH-based system, it has the value of who ran the command.

Now that the basics of the `yum history` command are out there, let's talk about the real magic. As part of the *Yum* system, it is possible to roll back a transaction. This doesn't remove the need for a backup but it can be useful if something suddenly doesn't work. To initiate a rollback is quite straightforward: just use `yum history undo 3` in our case to remove *Apache2*. That operation generates a transaction ID that you could use for the *Yum* history info. A word of caution, however…

This isn't guaranteed to work in every scenario. When looking to undo standard package installations, there should be no issue. We wouldn't dream of undoing (reverting) after a *Yum* update command, though, because there are too many moving parts and potentially significant changes. That kind of scenario is where VM snapshots come into play and make a revert to the previous working state easy.

As with most commands, a full breakdown of the command (and the other useful commands) can be found using `man yum`.

As a parting gift, another useful *Yum* command is `yum downgrade`. As the name suggests, this enables an administrator to downgrade the current version of a package to a lower version if required.

There are many save-my-backside tools built into *Yum/DNF*. It's just a case of knowing what is there and how it can be used effectively. LXF

Being able to see in detail who did what and when is a really useful tool that Yum provides.

```
[root@rockylinux-s-1vcpu-1gb-sfo3-01 ~]# yum history
ID     | Command line                                                                | Date and time    | Action(s)      | Altered
-------------------------------------------------------------------------------------------------------------------------------
     7 | install httpd                                                               | 2023-07-07 09:41 | Install        |   12
     6 | update                                                                      | 2023-07-07 09:35 | I, U           |   44
     5 | install -y droplet-agent                                                    | 2023-07-07 09:33 | Install        |    1 E<
     4 | -C -y remove firewalld --setopt=clean_requirements_on_remove=1              | 2023-05-13 21:45 | Removed        |   10 >E
     3 | -C -y remove linux-firmware                                                 | 2023-05-13 21:45 | Removed        |    2
     2 | install grub2-pc-modules grub2-pc -y                                        | 2023-05-13 21:45 | D, I           |    6 E<
     1 |                                                                             | 2023-05-13 21:42 | Install        |  451 >E
[root@rockylinux-s-1vcpu-1gb-sfo3-01 ~]#
```

HostPapa

Ruby P Jane tries this paternal web host provider that offers a range of services for both personal and business use.

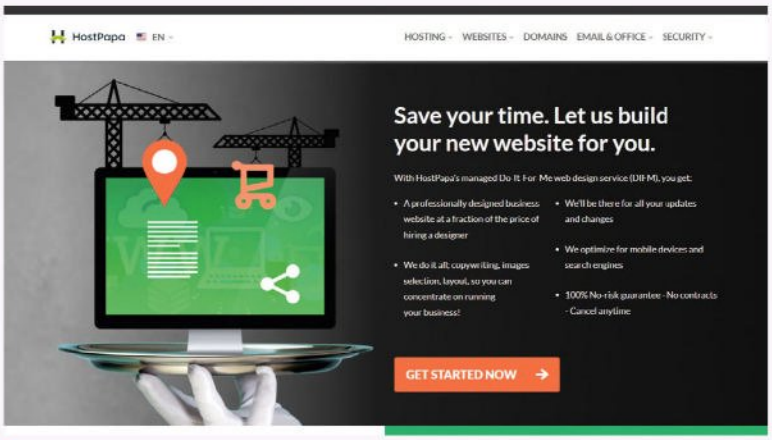
IN BRIEF

The most affordable plan starts at just $2.95 (around £2.30) per month and includes one website and 100GB of storage. If you need more features, such as unlimited websites and free domain registration, you can upgrade to a higher plan.

HostPapa has been around for more than a decade, providing web-hosting services to small and medium-sized businesses. It has grown to become a global leader in hosting solutions, serving over 500,000 customers in more than 100 countries.

Offering a variety of plans such as shared, VPS, reseller and WordPress hosting, it also provides domain registration, site builder tools, email hosting, ecommerce and more.

The web-hosting plan includes a free domain name, unlimited disk space and bandwidth, as well as personalised email addresses. It also comes with a user-friendly control panel, enabling you to manage your website easily from anywhere.


HostPapa
EN
HOSTING
WEBSITES
DOMAINS
EMAIL & OFFICE
SECURITY
Save your time. Let us build your new website for you.
With HostPapa's managed Do It For Me web design service (DIFM), you get:
A professionally designed business website at a fraction of the price of hiring a designer
We do it all: copywriting, images selection, layout, so you can concentrate on running your business!
We'll be there for all your updates and changes
We optimize for mobile devices and search engines
100% No-risk guarantee - No contracts - Cancel anytime
GET STARTED NOW

HostPapa's plans start out cheap but go up in price significantly upon renewal.

WordPress self-hosting includes a pre-installed environment, automatic updates, as well as enhanced performance through optimised settings. A managed WordPress plan goes further by providing dedicated customer support for WordPress issues. Here HostPapa takes care of updates, security patches and backups.

The reseller hosting plan is designed for businesses that want to provide web-hosting services to their own clients. With this, you can host multiple websites under one account. There's also a white-label solution, enabling you to customise the hosting experience for your customers under your own brand name.

HostPapa offers a do-it-yourself online Website Builder, which enables you to create, publish and optimise a beautiful website. It is user-friendly and doesn't require any technical skills, and offers a web store and ecommerce options for over 45 payment systems. Advanced HTML and CSS editing features are available for users who have more technical skills or specific customisation needs.

Papa's performance

Hosting is exceptionally reliable, with an uptime guarantee of 99.9%, and makes exclusive use of SSDs with its own CDN. It uses the LiteSpeed web server, which is supposed to be more optimised than a general LAMP server. Our uptime test via UptimeRobot that ran on HostPapa's main website for over a month didn't show a single trace of downtime and showed only a few major oscillations in response time.

Signing up, choosing a domain name and setting up email took only minutes. The dashboard layout is straightforward, icons and menus are self-explanatory, and video tutorials and a knowledge base are on hand.

HostPapa uses cPanel as its control panel, which is a popular choice among hosting providers. The cPanel interface has been tweaked to match its branding and add some useful features, such as *Softaculous*, a tool that automates software installations.

There is no monthly billing – just 12, 24 or 36 months, which seems inflexible in this day and age. On a more positive note, all plans offer decent value for money, with European, US and Canadian hosting options.

Help and support

HostPapa offers several ways to access its customer support team, while a knowledge base provides a comprehensive library of articles and video tutorials.

There's live chat support available 24/7, so you can get help whenever you need it. If you prefer to speak to someone over the phone, HostPapa's customer care centre is available at any time. There are email and fax support options, too.

Its support team is incredibly responsive and helpful. Whenever we had an issue with our website, we were able to reach someone within minutes. The dashboard is user-friendly and uptime impressive, while pricing is affordable compared to other service providers. LXF

VERDICT

DEVELOPER: HostPapa
WEB: www.hostpapa.com
PRICE: From $2.95 (approx £2.30) per month

FEATURES	7/10	EASE OF USE	9/10
PERFORMANCE	9/10	VALUE	8/10

Top-notch hosting web solutions at pocket-friendly pricing, with a 99.9% uptime guarantee. There is no free trial and automatic backup is charged.

» Rating 8/10

CREDIT: HostPapa

Cloudways

A competent host offering scalable cloud-based solutions that has **Ruby P Jane** loving what she sees.

IN BRIEF

With its robust infrastructure and intuitive user interface, Cloudways has become a go-to choice for anyone looking to harness the potential of cloud hosting. It offers a seamless experience and unmatched performance.

One of the standout features of Cloudways is its use of SSD-based hosting. With SSD drives, your site experiences sufficient performance improvements and reduced page load times. This is especially important for dynamic websites that rely on quick response times to engage visitors and keep them on your site. Cloudways also provides built-in advanced caches, including Memcached, Varnish, Nginx and Redis.

The platform is also PHP 7.4 and 8.0 ready. PHP 8 is known for its significant speed improvements over its predecessor, making your site faster and more efficient.

Unlike shared hosting, Cloudways offers a dedicated environment for your websites. This means your server resources are not shared with other users, enabling you to maximise server and app performance. And Cloudways takes care of server maintenance with auto-healing managed cloud servers.

Cloudways also includes pre-configured PHP-FPM, which dramatically speeds up your website and improves the load times of your PHP environment. To enhance the speed of communication between web servers and clients, Cloudways servers are HTTP/2 supported. There is HTTP/3 support with the Cloudflare enterprise add-on.

It is easy to secure your website with free one-click SSL installation using Let's Encrypt SSL. For enhanced security measures, there's IP whitelisting. With the help of Cloudways' technology partner, Malcare, you can protect your WordPress websites from malicious bots, brute-force login attacks and DoS attacks.

If you manage multiple websites, Cloudways allows you to launch more than 10 apps through one-click deployment. With this feature, you can deploy multiple WordPress and Magento sites on a single server, providing convenience and streamlining your workflow.

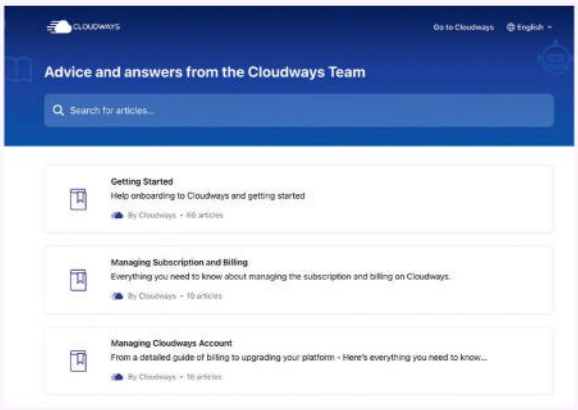
The latest versions of *MySQL* and *MariaDB* are on hand on all its servers. You can choose the database that works best for your application and switch between them as needed.

Support solutions

Cloudways offers enhanced service level agreements (SLAs) for those who require advanced support. You gain doubled live chat access and reduced online ticket response times to three hours for normal priority or 30 minutes for high priority. For those who require premium SLAs, Cloudways has even more efficient support options available. It also offers server customisation and configuration options. Whether you need assistance with deploying custom packages, caching configurations or server settings, Cloudways' support team can help.

Cloudways offers cloud hosting services from several companies, for which you can pay monthly or by the hour. With an hourly pay-as-you-go system, you're charged for what you actually use. With a monthly billing system, DigitalOcean's plans start at $14 (around £11) per month, with AWS at $38.56 per month and Google Cloud at $37.45 per month.

CREDIT: Cloudways


Advice and answers from the Cloudways Team
Search for articles...
Getting Started
Help onboarding to Cloudways and getting started
Managing Subscription and Billing
Everything you need to know about managing the subscription and billing on Cloudways.
Managing Cloudways Account

Its knowledge base and FAQ can answer many questions you may have.

You get the same features on all the plans, whether you go with DigitalOcean's lowest-priced plan or highest-priced from Google Cloud. Although there's no money-back guarantee, there is a three-day free trial, and you don't have to enter your credit card details to access it.

As a managed cloud-hosting platform, Cloudways offers a range of features and benefits that can cater to the needs of different users, such as developers, small and medium-sized businesses, and even larger enterprises. It provides access to a wide range of cloud infrastructure providers, including Amazon Web Services (AWS), Google Cloud Platform (GCP), DigitalOcean, Vultr and Linode, enabling you to choose the best one for you.

With its emphasis on simplicity and ease of use, Cloudways incorporates a user-friendly control panel that enables you to manage and deploy your applications with minimal effort, facilitating tasks such as server configuration, app installation and scaling resources to accommodate varying traffic demands. However, if you need email hosting and domain registration, Cloudways may not be the right choice for you. LXF

VERDICT

DEVELOPER: Cloudways
WEB: www.cloudways.com
PRICE: From $14 (approx £11) per month

FEATURES	9/10	EASE OF USE	8/10
PERFORMANCE	8/10	VALUE	7/10

Cloudways offers great features, but it lacks email hosting and domain registration, and imposes limitations on multisite SSL, making it a challenge for potential users.

» Rating 8/10

Cloud services have something to offer almost every kind of business. In fact, if you believe Amazon's hype, we're heading for a world where companies of all sizes run entirely on remote servers. Microsoft treads a little more cautiously, focusing on hybrid architectures, but either way, it seems certain that the cloud is integral to the future of many businesses.

The challenge is, it's not always easy to tell which cloud platform will be the best fit for a particular role in a particular organisation. That's partly because the different providers make broadly the same promises, enabling you to deploy services almost on a whim, and scale them up or down with minimal management.

But while the fundamentals might seem identical, as soon as you start to look in depth at the two biggest platforms – those being, without question, Amazon and Microsoft – you run into nuances and differences of philosophy that can have an enormous impact on either the service or the customer.

How did we get here?

Only around 15 years ago, most corporations were built largely on in-house hardware. This was a big expense: the capital footprint of a standard server, including the lifetime costs, right down to the ground rent for the rack in which the server would hide, could easily come to £250,000. And after that investment, most server CPUs spent the majority of their time sitting idle.

Then came the virtualisation revolution, which allowed businesses to work their IT investments harder by running a dozen virtual servers on the same piece of

JETTING OFF OUT OF THE CLOUD

A new trend is appearing in 2023: leaving the cloud. After years of big corporations (that happen to run cloud infrastructure) proclaiming the cloud is the best thing since sliced bread, it turns out it's not. Largely after years of investment and below-cost sales, companies that now feel they have a captive audience are starting to turn the screws.

According to CloudZero's 2022 report, only 40% of the respondents estimated their "cloud costs are about where they should be or lower", and in Anodot's 2022 report, 50% of IT managers said that "it's difficult to get cloud costs under control".

One real-world example is HEY (**https://world.hey.com/dhh/we-have-left-the-cloud-251760fb**), which explained that it took six months to migrate legacy services from the cloud. Even with the procurement of two pallets of Dell servers powering 4,000 vCPUs, it'll save the company $1.5 million per year. The chief technology officer said, "I still think the cloud has a place for companies early enough in their lifecycle that the spend is either immaterial or the risk that they won't be around in 24 months is high. Just be careful that you don't look at those lavish cloud credits as a gift! It's a hook."

But costs aren't the only reason to leave the cloud. IBM's 2022 Transformation Index shows that 54% of respondents agree that "public cloud is not secure enough". Concerns are driven by recurring vulnerabilities in cloud providers and the impact of cloud failures on the internet. Performance can also be an issue, but for more complex reasons, as cloud providers tend to provide better hardware than private owners, but local storage access advantages and lower local network latencies have unintended consequences.

Image credit: Memedozaslan/iStock/Getty Images Plus

AWS vs Azure
What's the difference?

Not all cloud providers are the same.
Steve Cassidy compares the two biggest platforms.

iron. The response was undeniable – we don't think it's an exaggeration to say that virtualisation was adopted more quickly than any other new technology in the history of computing. And it didn't just work for on-premises hardware, because it was the perfect fit for cloud services, supporting the growing web and mobile app marketplaces.

Amazon was quick to seize the opportunity, launching its Elastic Compute Cloud (EC2) service in March 2006. Microsoft followed up with Azure in February 2010. You might think that a four-year delay would put Microsoft at a huge competitive disadvantage, but at the outset there wasn't much of a sense of competition – or if there was, it was more between Amazon and traditional virtualisation solutions. In the end, VMWare put together its own cloud product, but it's still not what the company is best known for.

In fact, the timing of Microsoft's entry into the market was perfect. Following the credit crunch, businesses that had previously been cautious about migrating into the cloud became a lot more receptive to the sales pitch of elastic computing, and paying only a modest monthly fee for the resources they actually used. There were few other challengers; Google didn't get into general-purpose cloud computing until quite a few years later, while Akamai took little interest in small, commercial operations – as it continues to do.

That's not to say that Azure was an instant hit. While Amazon saw meteoric growth in the early 2010s, Microsoft was held back by technical limitations. Most notably, while Amazon had strong Linux support from day one, Azure launched as a Windows-only service, and took more than half a decade to mature into something sensibly OS-agnostic. That eventually changed in around 2018, when it was announced that Azure was now running fully on Windows Server 2012, supplying the exact same services as your in-house servers – including full Linux environments.

Does it really matter?

That historical recap may seem academic, but it very much shapes the services you can buy today. Amazon's approach has always relied on a very neutral hypervisor and data centre-scale network layer. If you want to run a Windows server on its platform, that's fine – although some of the more advanced configurations (such as server-to-server VPN tunnels in a Windows domain) might take a few updates and a bit of head-scratching to get working. The overall philosophy is one of a clean virtual architecture that replicates physical hardware.

> **NEBULOUS NUANCES**
> "As you start to look in depth, you run into differences of philosophy that can have an enormous impact"

The Azure model is more akin to deploying virtual Windows servers and standard services, rather than setting up a generic black box. If building a hybrid environment where virtual servers interoperate with real ones, that may be what you want. But it muddies the water regarding management and orchestration, as the virtual machine's resources won't square with those of the physical machine – and if you're interested in dynamic scalability, you need to keep an eye on both.

The best choice of cloud service therefore depends on your own technical histories and habits. To be clear, »

AMAZON VS AZURE HEAD TO HEAD

	AWS	Azure
Size	It was six times larger than the next 12 competitors combined in 2022, but its market share has fallen from 40% then, to 32% in 2023.	Reported four million servers, as of April 2021 – and growing. Reached 23% market share in 2023.
Big customers (reported monthly spend)	Netflix ($20 million); BBC ($9 million); Adobe ($8 million); Twitter ($7 million)	Verizon: ($80 million); LinkedIn ($41 million); Adobe ($40 million it uses both); Intel ($39 million)
Data centre locations	56 regions in 21 countries	60 regions in 140 countries
Service level	Refunds for downtime are defined per region	Better uptime when you pay for more expensive storage
Customer support	Strong phone-based support on SaaS services; weaker on raw compute and VMs	Relationships with corporate customers and emphasis on bigger projects; support mostly through partner and specialist suppliers
Pricing model	Four different levels, with cost rising with exclusivity of hardware	Three pricing strands based on overall available capacity on Azure; cheapest strand may be paused or dismounted when compute demand is high

you'd be pushed to find a job you absolutely could only do on one platform or the other. If you're accustomed to using remote Linux-based servers with lots of config files and obscure system security provisions, though, you'll feel at home with Amazon – whereas if you're a Windows shop, with wizards dominating your day-to-day activity and lots of scripting to join your various paid-for components into a coherent, singular service, you're likely to be more comfortable with Azure.

That'll cost extra, sir

The two big cloud services don't just embody different architectural philosophies – they have different approaches to pricing. Unfortunately, making sense of these isn't as easy as comparing two flat monthly fees, as it depends not only on the specific services you use, but how heavily you use them. What's more, pricing can vary, sometimes aggressively, in reaction to market conditions you can neither see nor negotiate with.

Still, for a rough and ready estimate, Amazon offers a pricing calculator at **https://calculator.aws**, while Microsoft's is at **https://azure.microsoft.com/en-gb/pricing/calculator/**. And you don't normally need to dig too deep to find out about pricing fluctuations, as both tend to keep the chatter in front of you all the time, along with all sorts of special bundles and deals. Amazon perhaps has a slight bias towards special deals on lumps of flat-rated hosting time and space, whereas Azure tends to talk up rapid-deployment solutions for specific jobs, such as AI-scanning your entire data lake for trends in your customer base.

Overall, managing your budget can be a little more complicated with Azure, as it involves a wider menu of bolt-on components and capabilities. This isn't necessarily a bad thing; if you want to run a few machines in a private role, linked back to your company office as a neat way of backing up your Active Directory config and data, and maybe a few important files, Azure can help every step of the way. It probably works out very cost effective at first, too, because Microsoft sugars the initial pill by giving away lots of free time and services to new relationships. When the next renewal comes up, however, things may get interesting *(see the Jetting Off box p.74 for comment)*.

Amazon offers a simpler service and pricing model – and while it does offer modular components, these don't always require you to spin up a complete VM. This makes it good value if you just want a static website or cheap and cheerful email configuration.

Do pause, however, to consider whether all your functions need to live in the cloud at all. Cloud repatriation is a big topic, with companies bringing services back in house after realising they were paying significant fees for things they could happily have continued running themselves. This was particularly easy to fall for in the early days, when Amazon still presented itself largely as an online bookshop; as most company credit cards don't limit spending on books, a lot of services were quietly moved into the cloud without explicit discussion or authorisation.

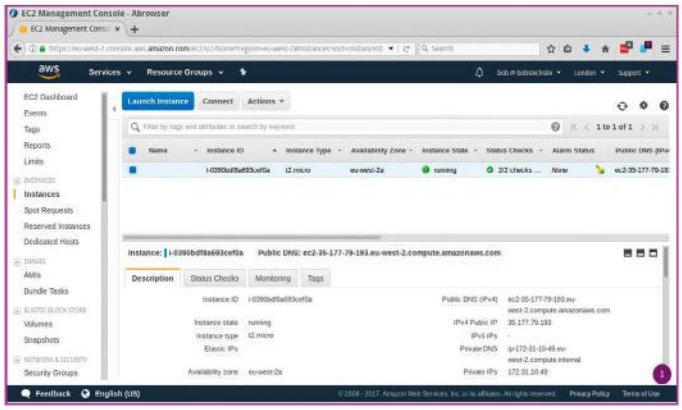
That's less the case with Azure, perhaps because it tends to deal with bigger corporate customers who negotiate a fixed-price contract from the get-go. That follows naturally from its long history of partner and reseller-orientated business, which Amazon never had.

Containers, VMs and Greeks

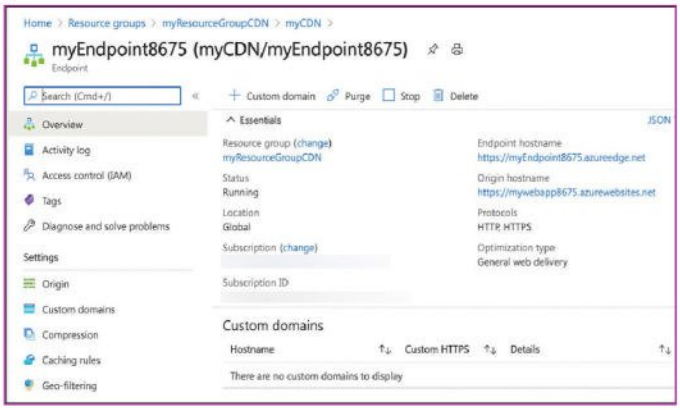
There's a lot of room in the cloud philosophy for different server usage patterns – and different hosting models. The least virtual part of the cloud is that concerned with serving up web pages and data, rather than doing lots of arcane computing. A lightweight web presence without too many customisations, shopping carts, blog engines and so forth is ideally suited to containerisation, rather than full-on virtualisation.

Container technologies have in general become a hot topic, with the main platforms being Docker, Butler and Kubernetes – Greek for the guy who steers the boat. The big idea is, rather than spinning up a complete virtual computer for each service you want to run, containers package up only the components needed to do a specific task. For many scenarios – such as a web server farm – these containers can be largely clones of one another, and any number of isolated containers can be run within a single OS instance. It's a huge increase in efficiency that could slash the running costs of your site, while retaining the possibility of scaling up if demand increases.

As well as saving resources, a container-based approach helps the orchestrator manage demand and



The control panel for a standard AWS EC2 compute instance – this one was running Nextcloud.



Part of the Microsoft Azure control panel for endpoint resource management.

system health. Even estimating load is complicated with a traditional hypervisor, as it requires you to peer inside the VM and monitor each virtual CPU – whereas discrete containers are much easier to keep an eye on.

If you want to take a containerised approach to your own software stack, that can be another differentiator between Amazon and Azure. The former happily lets you implement your own containers within the black-box VM model, but Microsoft is well ahead here, with a wealth of services designed to help you integrate containerised apps into your workflow.

Don't jump too hastily, though. For a small business, switching from full VMs to containers probably won't yield much benefit, simply because your traffic is unlikely to experience huge spikes in demand that need an instant scale-up in response. If a regular VM approach works out cheaper for the average month, it's probably worth sticking with that – add-on services will increase your ongoing costs, while the benefit may materialise only occasionally, or never.

A word of warning

There's a final caveat that applies to both AWS and Azure: one of the big attractions of cloud hosting is that it generally provides a degree of resilience that you won't get from an old tower server sitting in the corner of the office. However, there's no intelligent way for the host operator to back up your VMs – after all, your data and processes are normally kept private from the cloud operator, so it can't possibly tell when's a good (or bad) time to start a backup. For similar reasons, you can expect little help with general system health. It's down to you to work out whether your virtual machines are running stably, without configuration errors or infections – and to be prepared to take action as needed.

This means that any cloud-migration budget needs to take into account the right mix of virtual-machine support features, storage (both local and remote) and any necessary additions to regular system components. This is another area where Azure with its Microsoft roots can appeal to Windows admins, because it carries across the Distributed File System and File Replication Service data-protection features from the regular Windows Server platform. However, Amazon has its own AWS Backup service, too, which works across the company's whole range of Linux or other cloud services. LXF

» BEYOND THE BIG TWO

Google Cloud

Google got into the cloud game later than most but nonetheless has built a strong customer base. Google Cloud dedicates a lot of resources to security, with AI systems to detect and counter cyber threats. Its platform is famous for its high performance – Google has a lot of hardware and software expertise that it leveraged to design servers and server management software that work as fast as possible. The main complaint you'll hear about Google Cloud is that it's expensive, which is true. It costs more to use than rival platforms, but many people think the trade-off for high performance is worth it.

DigitalOcean

One of the most popular low-cost cloud providers, DigitalOcean doesn't offer as many sophisticated features as you'd find on rival platforms, but it focuses more on offering simple cloud computing features for an affordable price, and it does its job well. With just 15 data centres worldwide, it is much smaller than the networks of the likes of AWS and Azure. Its reliability and performance are good but not as good as what you can get on these rival platforms. The real attraction of DigitalOcean is those lower costs, which you get in exchange for fewer features and reduced performance.

OVHCloud

A French cloud computing provider focused on European customers, OVHCloud is the largest cloud-hosting provider in Europe and the third-largest globally based on physical servers. Choose the type you want and the region you want it to be in from VPS or dedicated servers. It's affordable compared to most rival platforms. You can rent dedicated servers for as low as $11 (around £8.50) per month or shared virtual private servers for as low as $4 a month. Support isn't as good – you need to pay a significant sum for dedicated human support, which puts it at a disadvantage.

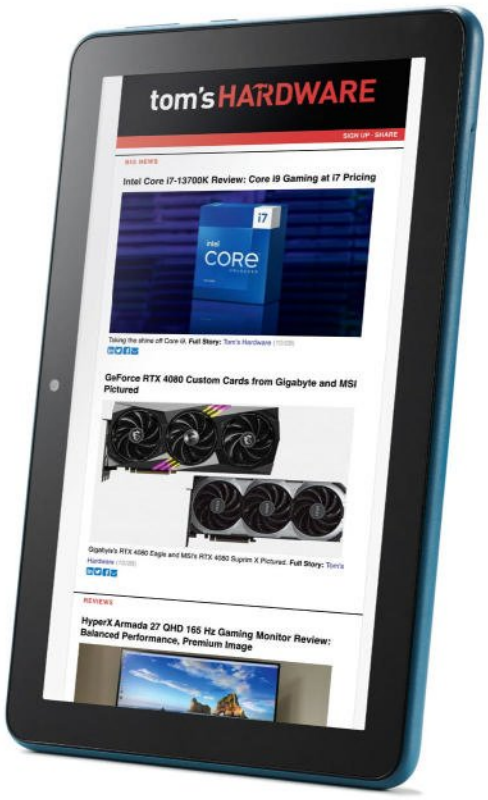
Image credit: AWS, Microsoft Azure

tom's HARDWARE

GET ALL THE ESSENTIAL BREAKING NEWS FOR THE TECH ENTHUSIAST!

No matter if you're building a PC, buying a laptop or learning about robots, Tom's Hardware has all the comprehensive knowledge you need.

Scan & Subscribe for free!


tom's HARDWARE
Intel Core i7-13700K Review: Core i9 Gaming at i7 Pricing
GeForce RTX 4080 Custom Cards from Gigabyte and MSI Pictured
HyperX Armada 27 QHD 165 Hz Gaming Monitor Review: Balanced Performance, Premium Image

NEW
The Python Book
Over 400 essential tips
OVER 2 HOURS OF VIDEO TUTORIALS
Learn to use Python • Program games • Get creative with Pi

GET READY TO BECOME A TRUE PYTHON EXPERT

Make Python work for you with tutorials on coding with Django, Flask, Pygame and even more useful third-party frameworks.

ON SALE NOW

FUTURE

Ordering is easy. Go online at:
magazinesdirect.com
Or get it from selected supermarkets & newsagents

HotPicks

Immich » G4Music » Cavasik » Footage » Login Manager Settings » Gear Lever » Nala » Alien Arena » Cartridges » Cubic » Filelight

Mayank Sharma rediscovered his old music collection and says that playing Robbie Williams in the background helped inspire this intro, if not the software selection.

PHOTO STORAGE

Immich

Version: 1.65.0
Web: https://immich.app

If you love the convenience of *Google Photos,* but aren't really comfortable placing your pictures in the hands of a third party, you have to give *Immich* a go. It's a self-hosted photo storage and sharing service that's pretty similar to the popular proprietary service, with both its intuitive web interface, and mobile app.

Immich is deployed as a *Docker* container, so you need to install *Docker* and the *Docker Compose* orchestration tool before you set up *Immich.* This shouldn't be an issue as all major distros have detailed instructions on installing *Docker* and *Docker Compose.*

You also need to make sure the computer you are setting *Immich* up on has at least 4GB of RAM, plus a dual-core CPU for an acceptable level of performance.

To ease the installation process, instead of manually using *Docker Compose* to deploy *Immich*, you can use the experimental script (that worked flawlessly in our test) to do the heavy lifting for you:

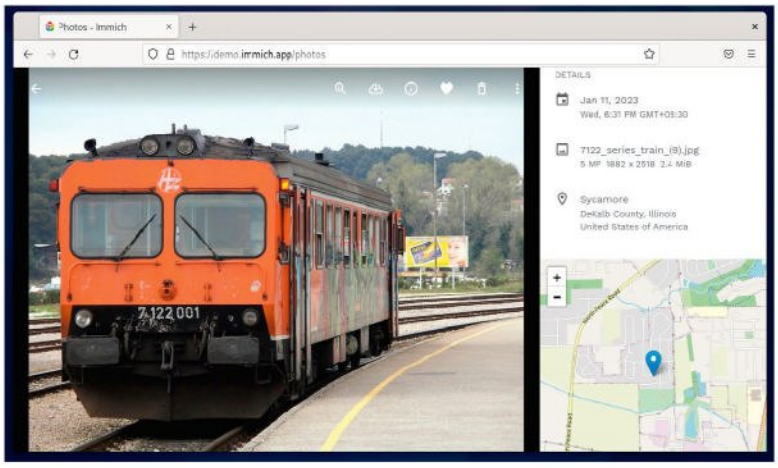
```
$ curl -o- https://raw.githubusercontent.com/
immich-app/immich/main/install.sh | bash
```

The command creates a directory named **immich-data** under the current directory and downloads and sets up your *Immich* instance automatically. The process might take some time depending on the resources at its disposal.

When it's done, it spits out the URL where you can access the *Immich* instance. For instance, if the IP address of the machine you've set *Immich* up on is 10.0.2.15, you can access your *Immich* web service at **http://10.0.2.15:2283**.

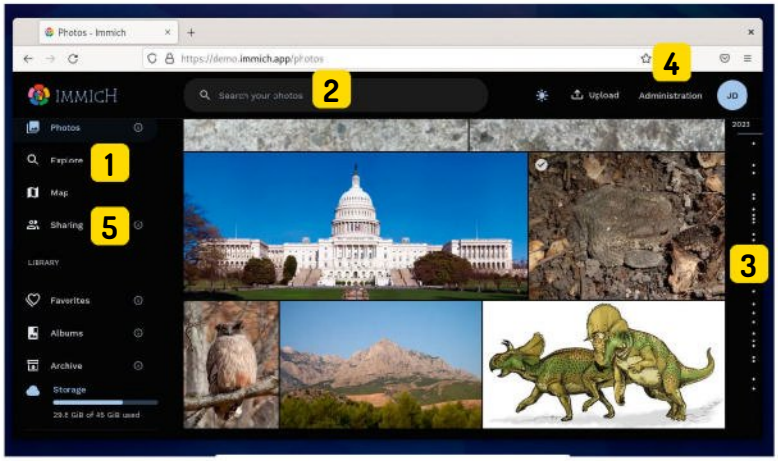
On initial login, you're asked to create an admin user. Later on you can use the admin user to add other users with restricted access to your *Immich* installation from under the Administration section. The default password for non-administrative users is password, which the users can change after logging in to the app.

After setting up the server, you can grab *Immich's* mobile app, available for both Android and iOS devices.



Immich's mobile app has an option to help you automatically back up the gallery from your mobile device into your Immich server.

LET'S EXPLORE IMMICH...



1 Explore your media
In addition to viewing your media, you can browse through your library using tools such as facial recognition and location data.

2 Search
Immich uses the Typesens search engine, which supports natural language processing to return accurate and relevant results.

3 Timeline
Use the timeline to quickly scroll through your library.

4 Administer the installation
Head to the Administration section to tweak your installation by adding more users and altering some essential settings.

5 Share
You can share your photos and videos with people in your network by creating shared albums, which can optionally be set to auto expire.

MUSIC PLAYER

G4Music

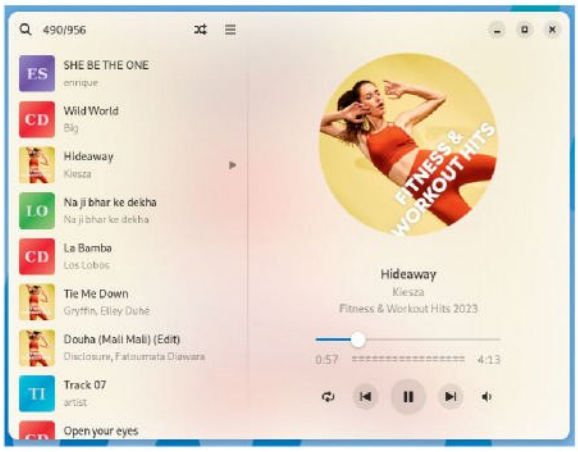
Version: 2.3 **Web:** https://gitlab.gnome.org/neithern/g4music

Old-school music aficionados like us have disks full of music that can often choke popular music players. If you, too, have a large library of music, what you need is a simple, straightforward client like *G4Music* that'll hum through the library without bothering with playlists or any curation.

G4Music is lightweight, both in size and features, but this is what makes it ideal for folks with humongous offline libraries. The app is available as a Flatpak and can be installed from FlatHub with `flatpak install flathub com.github.neithern.g4music`.

The app has a minimal but attractive and customisable interface. On launch, it automatically scans and loads all the audio it can find in the **~/Music** directory. The app supports virtually all popular audio file formats. You can point it to another directory in the app's Preferences. It also supports playback from remote locations with the help of protocols like Samba, thanks to GIO and *GStreamer*.

G4Music uses gaussian-blurred album artwork of the currently selected/playing track to decorate the app window, which makes the app look quite slick. You can turn this feature off by tweaking the preferences.



G4Music plays nicely on Gnome and can integrate with the MPRIS controls in the Gnome Shell notification tray.

By default, the app sorts the imported music in alphabetical order by title. One of the three buttons on the app's minimal interface helps you change this sort arrangement to list the songs by album, artist or by title of the song, or just shuffle it all up. There's also a search button that supports full text searches to help you locate songs.

Despite its minimalism, the app has several interesting features, such as background playback, as well as support for audio peak visualiser and gapless playback, and it supports several audio sinks including PulseAudio, PipeWire, ALSA, OSS and others. The app can also normalise volume levels with ReplayGain.

AUDIO VISUALISER

Cavasik

Version: 2.0.0 **Web:** https://github.com/TheWisker/Cavasik

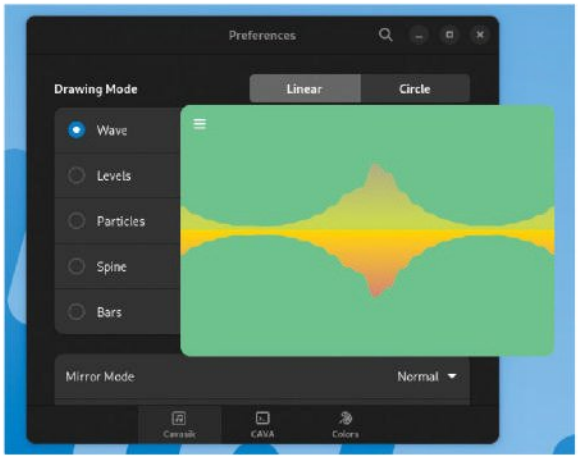
Another thing close to our hearts as old-school music aficionados is audio visualisers. These gizmos feature animated imagery that is generated and rendered in real time, in sync with the currently playing track.

Visualisers factor in attributes such as the volume of the music and its frequency spectrum to visualise the music. Although they aren't in vogue now, back in the day they were all the rage when you wanted something to stare at while listening to music.

Cavasik is based on the *CAVA* audio visualiser. It's available on FlatHub and can be installed with `flatpak install flathub io.github.TheWisker.Cavasik`.

The *CAVA* devs insist that their app isn't scientifically accurate, and is just designed to appear responsive and aesthetic with the current track. Pretty much in the same vein, *Cavasik* isn't designed to be of use to audio professionals, but it has plenty of bells and whistles to impress us audio enthusiasts.

The app supports multiple drawing modes including waves, levels, bars, particles and more. You can also draw these in circles or visualise them in one of the three supported mirror modes.



By default, the app doesn't show any window controls, for a cleaner interface, but these can be enabled from the Preferences window.

One of the best things about *Cavasik* is its plethora of customisation options. You can, for instance, set up to 10 colour linear gradients for both the background and foreground, and bundle them inside colour profiles.

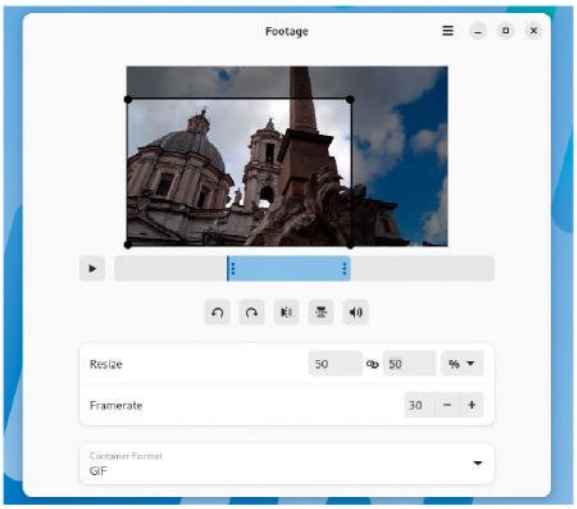
Because *Cavasik* is based on *CAVA*, you get access to most of the settings that make that app so good. For instance, you can control the number of bars in the visualiser, change the audio channels and add various levels of noise reduction or smoothing effects.

Cavasik's main window can be resized, and you can also make the application display sharp corners. You can also back up your configuration by exporting it, and even import someone else's.

VIDEO EDITOR

Footage

Version: 1.2 **Web:** https://gitlab.com/adhami3310/Footage



Footage is a nifty little video editor that offers some of the most common video-editing functions, but sadly it doesn't do batch processing.

Linux users have access to lots of quality open source video editors, such as *Kdenlive, OpenShot, Shotcut* and more. While they are good at what they do, they might be overkill for times when all you need to do is reduce the size of your video, or clip portions of it into a GIF. If your video-editing needs are few and frugal, you can get good mileage from *Footage*.

Unlike its well-known counterparts, *Footage* does only a handful of things. You can, for instance, use it to cut clips, change the frame rate, resize video, export it in a different format and so on.

As you can tell ,these features won't impress a professional video editor, but they're plenty for an occasional editor looking to cut out distracting friends from their wedding anniversary footage or reduce the size of their holiday movie so they can send it as an email attachment.

Footage is available as a Flatpak and can be installed from FlatHub with `flatpak install flathub io.gitlab.adhami3310.Footage`.

True to its nature, the app has a straightforward interface. Use the Open File button to select the video file you want to edit. Once it is loaded, you can use the controls to edit it as per your needs.

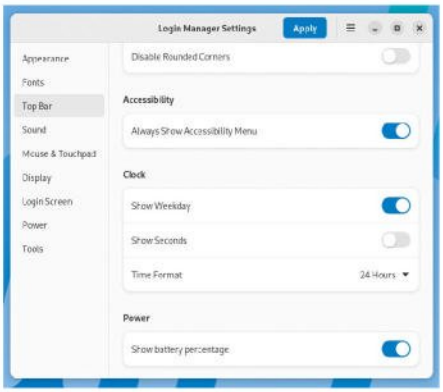
You can resize both the video as well as its length by making adjustments in the preview window and the timeline. Similarly, you can reduce the size of the video, either by the length and width percentage, or by pixels. In the same vein, you can also increase or decrease its frame rate as per your requirements.

After making the tweaks, you can export the new video in any of the popular container formats, including WebM, MP4, MKV or even as a GIF.

LOGIN CUSTOMISER

Login Manager Settings

Version: 3.2 **Web:** https://gdm-settings.github.io



Once you've tweaked the login manager as per your needs, you can use the app to export the settings and import them into a different Gnome-based distribution.

It took some time, but we've got accustomed to the ways of the reimagined Gnome desktop. However, if there's one thing that still doesn't sit well with us, it's the lack of customisation options. Sure, the desktop has an entire extension infrastructure to help us mould the desktop as per our needs, but the desktop environment leaves quite a lot to be desired when it comes to making one-off tweaks.

One area where the stock Gnome release doesn't offer any customisation options is the desktop's login screen, the Gnome Display Manager (GDM). If the GDM is something you'd like to tweak, you should get your hands on the *Login Manager Settings* app.

Despite its non-aligned name, the app is built specifically for Gnome and enables you to change various GDM settings. You can, for instance, use the app to apply themes, change the background of the login screen, change the cursor theme and icon theme, apply night and light settings, and more.

It is available both as an AppImage as well as a Flatpak, although the developers themselves say that the AppImage may not work properly under certain conditions. This is why it's best to install the app as a Flatpak from FlatHub with `flatpak install io.github.realmazharhussain.GdmSettings`.

The app is fairly intuitive, with customisation options organised into nine sections: Appearance, Fonts, Top Bar, Sound, Mouse and Touchpad, Display, Login Screen, Power and Tools. You can jump through the sections and scroll through the options, and tweak them as you wish. Some settings are distro-specific, however, and may not have any impact on your distro. After tweaking the settings, click Apply. You must then log in again or reboot for the changes to come into effect.

APPIMAGE MANAGER

Gear Lever

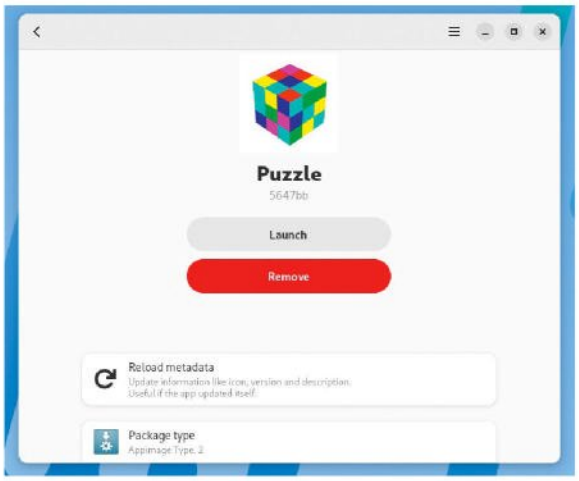
Version: 1.0.8 **Web:** https://github.com/mijorus/gearlever

AppImages, and Flatpaks for that matter, have streamlined software installation across Linux distros, taking away the complexities of dealing with different software installation tools and packaging formats.

But that's not to say these packaging formats don't have their shortcomings. The biggest, as far as AppImages are concerned, is their inability to integrate with the distro's Applications menu. That's where *Gear Lever* comes in, which helps integrate AppImages into your app menu with just a few mouse clicks.

Interestingly, *Gear Level* is available on FlatHub and can be installed with `flatpak install flathub it.mijorus.gearlever`.

The app opens up with a blank slate. You can either drag and drop an AppImage into the interface to manage it or click the + icon to select an AppImage using the file manager. After importing an app, you have to toggle a checkbox to confirm that the AppImage is legit. *Gear Lever* also displays the MD5 checksum of the file to help you verify the app's integrity.



After you've added an app to the Application menu, Gear Level displays a button that enables you to drop the app from the menu.

Once an app is added, *Gear Lever* gives you a couple of buttons: one to launch the app and another to add it to the distro's Applications menu. Use the first to check the app works as advertised before smashing the second one to list the app in the menu.

If you're like us and you've got a bunch of AppImages, you can use *Gear Lever* to house all of them in a folder of your choosing. By default, it houses all AppImages it manages under **~/AppImages**.

You can change the default location from the app's Preferences menu. By default, *Gear Lever* helps you save space by moving the AppImage into the default folder, although you can ask it to keep the original file and instead create a copy in the destination folder.

PACKAGE MANAGER

Nala

Version: 0.13.0 **Web:** https://gitlab.com/volian/nala

Apt does a nice job of package management on Deb-based distros, but it does leave some room for improvement. And *Nala* manages to nicely squeeze in to fill that role.

Nala, which was inspired by Fedora's *DNF* package manager, offers easier-to-read output and increased speed, thanks to parallel downloading.

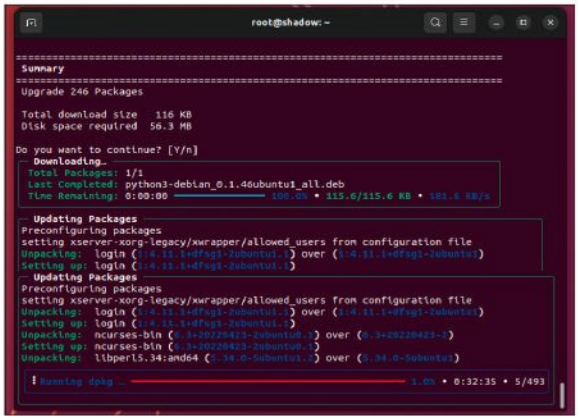
It is available in the official Debian Testing and Ubuntu 22.04 repos, and can be installed with `sudo apt install nala`.

If you use an older version of Ubuntu, you can still install and use *Nala* successfully by installing it via third-party repositories. As a first step, access and add the Volian Scar repositories to install *Nala* on Ubuntu:

```
$ echo "deb [arch=amd64,arm64,armhf] http://deb.volian.org/volian/ scar main" | sudo tee /etc/apt/sources.list.d/volian-archive-scar-unstable.list
```

Then grab the GPG verification key with:

```
$ wget -qO - https://deb.volian.org/volian/scar.key | sudo tee /etc/apt/trusted.gpg.d/volian-archive-scar-unstable.gpg > /dev/null
```



Unlike traditional Apt, Nala is more verbose and clear. For example, 'sudo nala upgrade' lists all packages that'll be installed and upgraded, as well as the scope of the upgrade.

And, finally, install *Nala's* legacy version:

```
$ sudo apt update
$ sudo apt install nala-legacy
```

Nala is a drop-in replacement for *Apt*, and is compatible with the majority of *Apt* commands. There are also some new commands that can be handy. One such command, `sudo nala fetch`, helps you to find the fastest mirror to speed up package installations.

Furthermore, *Nala* offers a history (see *Administeria p71*) function, inspired by Fedora's *DNF* package manager, which recaps the changes made during an update, and also helps you roll them back.

The command `sudo nala history` displays the most recent actions. Each action is listed along with an ID, which you can then use to undo the action, such as `sudo nala history undo <ID>`.

FIRST-PERSON SHOOTER

Alien Arena

Version: 7.71.4
Web: https://alienarena.org

Some Alien Arena maps contain hazards, such as deathrays, piranhas and lava, that you must avoid but can also use to kill other players.

If you like first-person shooters as much as we do, you'll love *Alien Area.* Nearing two decades on, the game's sci-fi theme and the *Quake*- and *Unreal Tournament*-inspired deathmatches are still as popular, as are its plethora of configurable parameters, tons of maps and multiple game types.

The game is available in the official repos of several distros, such as Ubuntu, but to ensure you have the latest version, download it as an official Flatpak with `flatpak install flathub org.alienarena.alienarena`.

Alien Arena focuses mainly on online multiplayer action, although it does have an offline practice mode to help you hone your skills against bots. To play online, you have to first browse the list of servers, each of which runs different maps and gameplay modes.

The game supports several gameplay modes, the most popular being Capture the Flag and Deathmatch, along with several others such as Deathball, Team Core Assault, All-Out Assault and more.

It takes place in environments set in a futuristic universe, and you get the option to choose from one of the several humanoid and alien characters as your avatar. The size, visibility and profile of each player character is roughly the same, and there is no advantage in selecting one over the other.

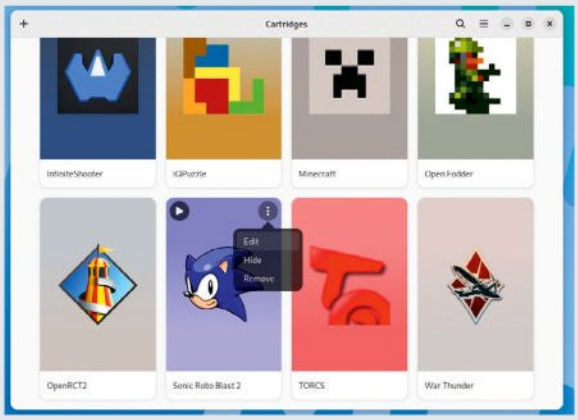
When you join a server, you have the option to join one of the two competing teams in the map that's running. Each player is given a blaster upon spawning, and while the human and alien weapons look different from each other, they are identical in their behaviour.

Before you begin playing, remember to visit the game's configuration options. From there you can customise gaming controls for both the mouse and the keyboard. Besides these, you can also alter a variety of settings that impact the game's performance, graphical quality, visibility and what is displayed on the screen.

GAME LAUNCHER

Cartridges

Version: 2.0.4 **Web:** https://github.com/kra-mo/cartridges



Unlike dedicated launchers, Cartridges doesn't give you options to optimise gaming experience, but is helpful if you can't be bothered to remember which game belongs to which launcher.

Having to switch between multiple launchers to start various games is a pain, especially if you, like us, have a large and diverse gaming library.

Cartridges is one of many available options that help unify all your games from different launchers. It has support for importing games from popular gaming launchers including Steam, Lutris, Heroic and more. The tool can also import games from Bottles, which enables you to run Windows-only games on Linux.

Cartridges is available as a Flatpak and you can easily install it on your Linux distro from FlatHub with `flatpak install flathub hu.kramo.Cartridges`. Once installed, you can launch the application from your distro's Applications menu.

By default, the installed games do not show up in the application and it looks barren. To get started, click the + icon on the top-left and then select the Import Games option to allow the app to scan your computer and add any installed games.

If, for instance, you have stored your Steam Library at multiple locations, *Cartridges* automatically detects them all. That said, you do get the option to add games manually, if you have installed them in a custom location.

The app tries to ape the appearance of a local video game portal by listing all supported games inside a unified interface. By default, the games are sorted in alphabetical order, but you can also sort them in reverse alphabetical order, or on the basis of when they were last played, and more.

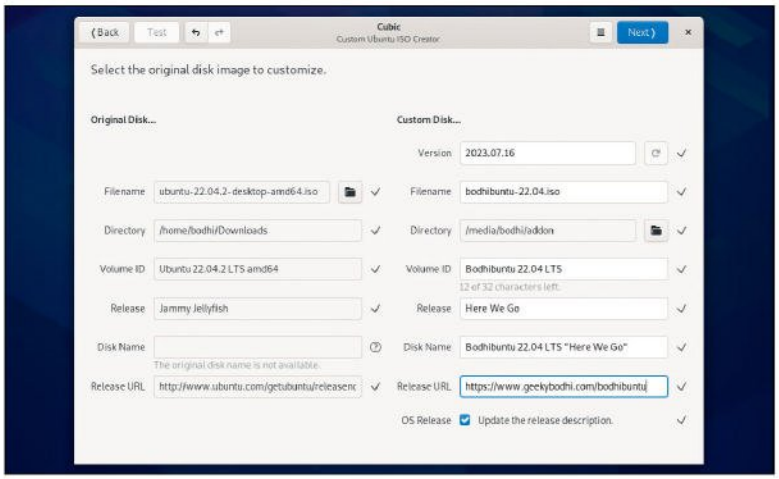
Cartridges gives you a search bar to rummage through your library, too. There's also the option to hide a title from the list, as well as a menu item to only show the hidden games.

One of *Cartridges'* nice touches is its ability to download game covers, including animated ones, from SteamGridDB, which makes it look quite spiffy.

CUSTOM DISTRO CREATOR

Cubic

Version: 2023.05.83 **Web:** https://github.com/PJ-Singh-001/Cubic



Cubic is fairly straightforward to operate, but the project does host a comprehensive guide on its project page to hold your hand through the process.

Ubuntu is one the favourite distros for creating derivatives, and *Cubic* is one of the tools that can help you make one for yourself.

Unlike some other tools for the job, *Cubic* is actively maintained and offers a lot of flexibility to roll your own custom Ubuntu-based distro. You can, for instance, use it to customise the packages in your distro, swap kernels, drape a custom wallpaper, and a lot more.

Cubic runs on all recent Ubuntu and derivatives. Enable the Universe repo before adding the *Cubic* PPA:

```
$ sudo apt-add-repository universe.
$ sudo apt-add-repository ppa:cubic-wizard/release
```

Then update your installation and install *Cubic*:

```
$ sudo apt update
$ sudo apt install cubic
```

Cubic has a wizard-driven interface, and when you start the app for the first time, you're asked to select a project directory, followed by the details for your custom distro, such as the source ISO, name and more.

It then extracts the ISO and gives you a chroot environment where you can execute commands. If, for instance, you want your custom ISO to have the freshest software, run `sudo apt update && sudo apt upgrade`.

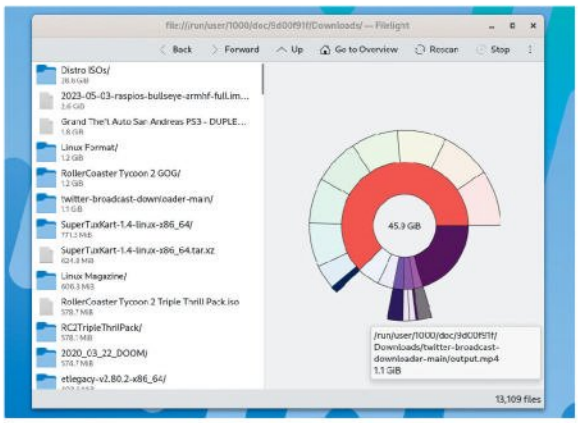
You can then use *Apt* to add or remove software. You can also use this chroot environment to update the list of software sources. Similarly, you can also copy any files and folders you want inside your custom distro.

If you've installed additional kernels besides the default one bundled in the ISO, *Cubic* gives you the option to select it as the default one for your custom distro. After you've run through the wizard, *Cubic* generates the ISO of your custom distro and also lets you test it using *Qemu*.

DISK SPACE MANAGER

Filelight

Version: 23.04.3 **Web:** https://apps.kde.org/filelight/



By default, Filelight only focuses on files that are taking up a large amount of disk space, but you can ask the app to display small files as well.

While a rolling stone gathers no moss, the more you use a distro, the more files it accumulates. Over time, these unnecessary files clog up your partition and slow down disk operations.

Finding disk space usage is no big deal in Linux, thanks to the venerable `du` command that can calculate and summarise disk space usage in a jiffy. But if you want something that's just as good but graphical, there are few better options than *Filelight,* which displays disk usage results in a very distinctive coloured radial layout.

Although designed for KDE, *Filelight* is available as a Flatpak, which makes installing it even in non-KDE distros a breeze. You can grab it from FlatHub with `flatpak install flathub org.kde.filelight`.

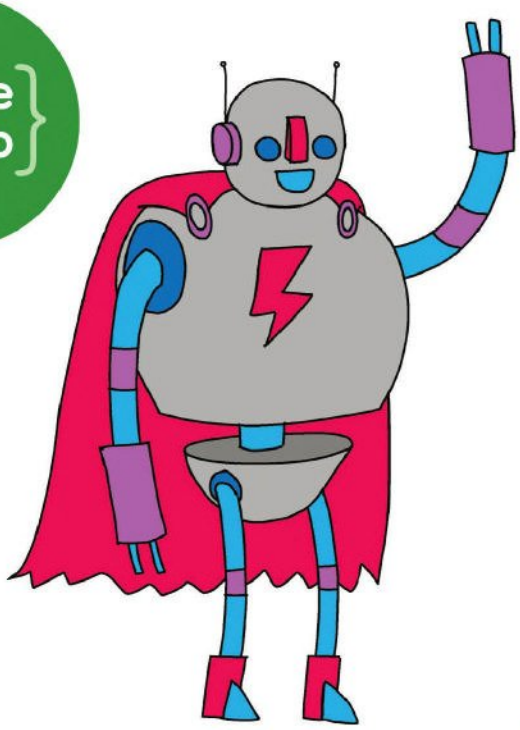
You can use the app to either visualise the complete filesystem or analyse the contents of a specific folder to identify hot spots or files and subdirectories that are eating up the most space on your disk.

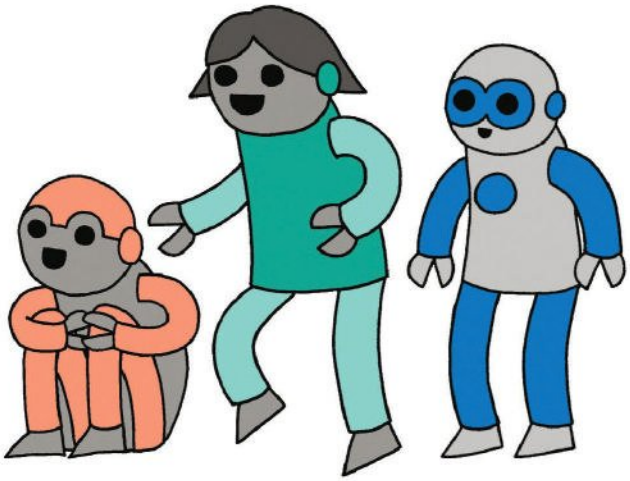
The app graphically represents the space used by the files inside the selected folder or your filesystem as a set of concentric segmented rings. Each segment in the radial layout is represented by different colours. Brightly coloured segments are folders, while grey segments are files.

All segments are labelled, and you can hover over each of them to view details about the specific files. When you click on a tile, the app offers to open the selected file for you. You can also right-click on a tile, which brings up a context menu that among other options gives you the ability to open a file in the terminal.

The app supports several different colour schemes, with the default one known as Rainbow. You can switch to a different colour scheme by heading to *Filelight*'s Configuration section and switching to the Appearance tab. LXF

{code club}





Can you help inspire the next generation of coders?

Code Club is a nationwide network of volunteer-led after school clubs for children aged 9-11.

We're always looking for people with coding skills to volunteer to run a club at their local primary school, library or community centre for an hour a week.

You can team up with colleagues, a teacher will be there to support you and we provide all the materials you'll need to help get children excited about digital making.

There are loads of ways to get involved!

So to find out more, join us at **www.codeclub.org.uk**

LXF ADVENTURE

Credit: https://github.com/azuregate

Part Three! Don't miss next issue, subscribe on page 18!

Text adventure combat mechanics

Often found making love and not war, this month **Nate Drake** takes our interactive text adventure down a dark, violent path.

OUR EXPERT

Nate Drake wants to dedicate this series to his dad who once told him that no one was ever going to pay him to sit around and play video games.

If you've been following the previous tutorials on creating your own Python text adventure you should now have a script capable of moving the player around any number of rooms and interacting with items.

This is fine but most dungeon crawlers usually involve some element of risk. Of course, as the code stands, it's quite possible to create a room to trap a player by removing all the **exit** attributes, but most games usually give the hero a certain quantity of health or hit points, which they have to guard carefully.

Hit me up

In order to make our hero sufficiently mortal, we first need to update the **Player** class in order to introduce the hit points attribute:

```
class Player:
  def __init__(self, name, currentroom, keyring, hp, inventory):
    self.name = name
    self.currentroom = currentroom
    self.keyring = keyring
    self.hp = hp
    self.inventory = inventory
```

This also means we need to assign a certain number of hit points (hereafter known as HP) to our player at the start of the script:

```
player = Player(player_name, room1, [], 10, [bread])
```

Note in this case we've decided to assign the player 10HP but feel free to change this according to how risky or easy you want to make your adventure.

Inheriting items

Our player now has a certain number of HP but we haven't done anything particularly remarkable, given that nothing in the game can hurt or heal them.

We could update the **Item** class to include an attribute to do this but this is unwieldy, plus it makes it difficult to tell the difference between different types of items. Fortunately, Python has a neat workaround for this by enabling you to create subclasses. A subclass has the same attributes as the parent class but also allows you to add other attributes of your own.

In this case, we'll use this handy Python feature to create a new object called **StatItem**:

```
class StatItem(Item):
  def __init__(self, name, itemdesc, updroomdesc, portable, revealsitem, usedin, usedesc, removesroomitem, addsroomitem, useroomdesc, disposable, hp_change):
    super().__init__(name, itemdesc, updroomdesc, portable, revealsitem, usedin, usedesc, removesroomitem, addsroomitem, useroomdesc, disposable)
    self.hp_change = hp_change
```

As you can see, this is based on the original **Item** class but introduces a new attribute, **hp_change**. This indicates the number by which your HP changes if the **StatItem** is used in-game. (Note that this can be an negative value, so **StatItem** covers both helpful and harmful items).

If you download the test script *(see Quick Tip, left)*, you'll see that the item **bread**, which is in the player's inventory, has been tweaked to become a **StatItem**:

```
bread = StatItem("Bread", "A loaf of fine Dwarfish Sabmel bread.", None, True, None, 0, "You eat the Sabmel bread. It's very refreshing.", None, None, None, True, 2)
```

In this case, the idea is that if the player eats the bread, they gain 2HP. Of course, in order to do this, we need to insert some code at the start of the **trytouse** procedure:

```
if isinstance(inventory_item, StatItem):
  player.hp += inventory_item.hp_change
```

Take a moment to review the example code at this stage. Hopefully, you can now see why it makes so much sense to use the **StatItem** subclass. The **trytouse** procedure now checks the item type to see whether it's a **StatItem** and modifies your HP accordingly. It then continues with the rest of the **trytouse** procedure as normal, for instance by printing the **usedesc** of what happens when you use the item.

QUICK TIP

You can download an example of the code, including items that can change the player's hit points, from **https://github.com/azuregate/lxfpythontextadventure/blob/main/nextroomstatscript1.py**.

Before running your modified script, it might be helpful for the player to be able to check their HP. This is best done by updating the **listinventory** procedure:

```
def listinventory():
  print(player.name)
  print("HP:",player.hp)
  print("You are carrying:")
  if not player.inventory:
    print("Nothing.")
  else:
    for item in player.inventory:
      print(item.name)
```

Take some time now either to run the example script **nextroomstatscript1.py** or your own modified version of the script. First, type `inventory` to see the readout of the player's HP. Next, enter `use bread` to consume it and gain 2HP. (The bread is already in the player's inventory in this case.) Finally, run `inventory` again to see your hero now has a hearty 12 HP.

Taking risks

So far the game has the ability to create items that can change the player's HP when used. You've seen how the bread nourishes your hero for +2HP. But if you feel like experimenting, feel free to swap it out for something harmful. For example:

```
apple = StatItem("Apple", "A small crabapple.", None, True, None, 0, "You eat the apple. It's very sour and makes you queasy. Lose 1 HP", None, None, None, True, -1)
```

If you run this code, eating the apple causes the player to lose 1HP. This is fine for items already in your inventory, but what about interacting with items in the room?

In some adventures, the act of trying to touch or move an item can sometimes trigger an event like a trap, which can cause the player to lose some HP.

Take a moment to review the example script, which is based on **room5 (West Chamber**). If you've been following this series, you'll know in the main adventure script this room contains a large metal grille, which you can't move. This is because the **grille** item's **portable** attribute is set to **false**. Instead of just making the grille immovable, though, what if it injures the player if they try to budge it?

In order to do this, we simply need to tweak the **grille** to be a **StatItem**:

```
grille = StatItem("Grille", "The iron grille is old and rusty but you see some sharp edges.", None, False, None, None, "You grasp the grille and try to move it but only manage to scratch your hand. Lose 3 HP", None, None, None, False, -3)
```

Take a moment to review this code. In the first instance, if the player uses the `LOOK` command to examine it, they'll see an oblique warning that there are some sharp bits of metal. The **usedesc** has also been updated to explain that by trying to move the grille your hero's been injured. (In this case, they lose 3HP.)

As things stand, typing `USE GRILLE` in the game has no effect, because the `USE` command only applies to items that are in your inventory. In order to make this work, you need to update the start of the **trytotake** command:

```
    if isinstance(room_item, StatItem):
      player.hp += room_item.hp_change
      print(room_item.usedesc)
      return
```

This nifty bit of code is very simple to follow. First it checks that the item in question is indeed a **StatItem** and if so updates the player's HP by its **hp_change** value. It then prints the **usedesc** to show what happens when this particular **StatItem** is used. We've used the **return** command here to exit the procedure, because there's no need to check whether items like these can be added to the inventory.

Of course, if you want to modify **trytotake** to allow items that modify the player's HP, then can be picked

QUICK TIP

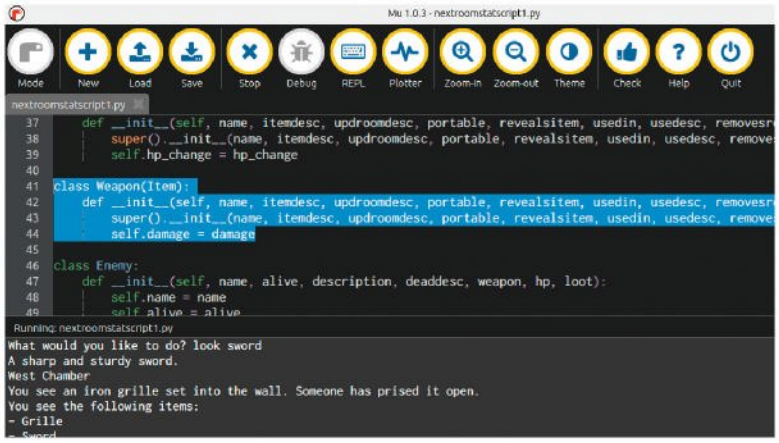
In programming circles, the mechanism of deriving new classes from existing ones is known as inheritance. The first recorded example of this is with the Simula programming language in 1969, around the same time text adventures were first being created.

» CODING COMBAT

If you've followed the steps in this tutorial, you'll see that it allows you to create weapons and enemies. For our Python text adventure, combat mechanics are managed by the procedure **fight**:

```
def fight(enemy_name):
  current_room = player.currentroom
  enemy = None

  if player.weapon == None:
    print("You can't fight without a weapon!")
    return

  for room_enemy in current_room.enemies:
    if room_enemy.name.lower() == enemy_name.lower():
      enemy = room_enemy
      break
  if enemy and enemy.alive:
    print(f"A battle begins with the {enemy.name}!")

    while enemy.alive and player.hp > 0:

      player_damage = player.weapon.damage
      enemy.hp -= player_damage
      print(f"You hit the {enemy.name} with your {player.weapon.name}. It causes {player_damage} damage.")


      if enemy.hp <= 0:
        enemy.alive = False
        print(f"The {enemy.name} has been defeated!")
        break

      enemy_damage = enemy.weapon.damage
      player.hp -= enemy_damage
      print(f"The {enemy.name} hits you with its {enemy.weapon.name}. It causes {enemy_damage} damage.")

      # Check player's HP
      checkhp()
    if player.hp <= 0:
      print("You have been defeated! Game Over.")
      exit(0)
  else:
    print("There is no such enemy here.")
```

This code checks the player is armed before allowing them to fight. The player and enemy then take turns hitting each other. The HP lost is determined by the type of weapon you or your enemy are using. Naturally, this isn't the most sophisticated method of fighting, so you may want to add a little unpredictability via Python's `random` command.



Weapons are a subclass of Item. There's an extra damage attribute to determine how many HP damage they cause.

up, feel free to do so. This is your world, as the great Bob Ross would have put it.

Take a moment to run the example script or your modified script and enter `TAKE GRILLE`. You'll see that the player loses 3HP. You can use `INVENTORY` to confirm the HP are deducted correctly each time.

Proof of life

If you followed our advice in the previous section to have the player repeatedly injure themselves on the grille, you'll have noticed your hero's HP going down way past zero, even as the game continues.

This is because, as things stand, the game doesn't check your HP each time you carry out an action. This is important to end the game if your HP reaches zero. Most games also place an upper limit on the number of HP the player can have to stop them claiming bonuses repeatedly.

In order to manage this, we need to create a new procedure, **checkhp**:

```
def checkhp():
 if player.hp > 10:
  player.hp = 10
 elif player.hp <= 0:
  print("You have been killed!")
  exit(0)
```

Simply insert `checkhp()` immediately underneath `while True:`.

This means that every time the player makes an action, the game checks their HP. If they do something that would take their HP over 10, such as eating the bread while already healthy, their HP remains at 10. If their HP is less than or equal to zero, the 'Game Over' message is displayed and the program exits.

QUICK TIP

You can download an example of all the code listed in the tutorials so far (including combat and looting bodies) from https://github.com/azuregate/lxfpythontextadventure/blob/main/adventure3.py.

Whip out weapons

At this point, your game can support items that affect the player's own HP, but no text adventure would be complete unless it gave players a chance to be armed to the teeth.

Still, in the interests of fairness and practicality, it makes sense for the player to only have and use one weapon at a time.

For this section, we recommend you download and read through the second example script before you start coding. It's available at **https://github.com/azuregate/lxfpythontextadventure/blob/main/nextroomstatscript2.py**. Begin by adding a **weapon** attribute to the **Player** class:

```
class Player:
  def __init__(self, name, currentroom, keyring, hp, inventory, weapon):
    self.name = name
    self.currentroom = currentroom
    self.keyring = keyring
    self.hp = hp
    self.inventory = inventory
    self.weapon = weapon
```

Don't forget to modify the **player** object but make them unarmed for now:

```
player = Player(player_name, room1, [], 10, [bread], None)
```

As weapons have special attributes, it's best also to create them as a subclass of **Item**:

```
class Weapon(Item):
 def __init__(self, name, itemdesc, updroomdesc, portable, revealsitem, usedin, usedesc, removesroomitem, addsroomitem, useroomdesc, disposable, damage):
  super().__init__(name, itemdesc, updroomdesc, portable, revealsitem, usedin, usedesc, removesroomitem, addsroomitem, useroomdesc, disposable)
  self.damage = damage
```

As with **StatItem**, the **Weapon** subclass inherits all the **Item** attributes and also adds another – in this case, **damage**. This is just a numerical value that reflects the number of HP the weapon deducts when used in combat.

Now that's created, have some fun by creating your very first weapon. If you're stuck for inspiration, take a look at the one we created in the example script:

```
sword = Weapon("Sword", "A sharp and sturdy sword.", None, True, None, 0, "You swing the sword around with great skill.", None, None, None, False, 2)
```

Weapons can be examined via `LOOK` in the usual way. If you enter `USE`, you also see text implying you wave it about in the air. This is because weapons are automatically deployed in combat, so you don't have to type in a separate command (see below). The final attribute indicates the HP damage (in this case, 2).

Next we need to modify the **trytotake** command so that if a player picks up a weapon, it's added to their **weapon** attribute, not their backpack:

```
if isinstance(room_item, Weapon):
 print("You have taken the", room_item.name + ".")
 player.weapon = room_item
 current_room.items.remove(room_item)
 return
```

Note that as the code stands, picking up a new weapon removes the old one permanently. This makes sure the player can only use one weapon at a time but it also runs the risk the player could pick up an inferior weapon. You may want to modify this code to check for this, or simply have them drop the old weapon in the room where they pick up their new one in case they change their mind.

In order to help players keep track of their weapons, consider modifying the **listinventory** command as shown in the example script:

```
 if player.weapon != None:
  print("Current weapon: ", player.weapon.name)
 else:
  print("Current weapon: None")
```

Run the game and enter `INVENTORY` before and after picking up the sword to check that it's equipped.

Enter the enemy

If you took the time to run the example script or are working on your own version of a text adventure, by now you're able to create and pick up weapons, but there's no chance to use them.

Although not all text adventure games include combat mechanics, those that do usually introduce enemies whom the hero needs to defeat in order to continue. Sometimes they hold weapons or items that are useful to the player.

To get started with this, scroll to the top of your code to create a new **Enemy** class:

```
class Enemy:
  def __init__(self, name, alive, description, deaddesc, weapon, hp, loot):
    self.name = name
    self.alive = alive
    self.description = description
    self.deaddesc = deaddesc
    self.weapon = weapon
    self.hp = hp
    self.loot = loot
```

If you've been following the coding tutorial so far, this shouldn't be too hard to understand. Enemies have a name, then a Boolean (true/false) attribute to show whether they're alive. There's a description text for when they're both alive and dead. They have a weapon and HP for combat. The **loot** attribute is to store items found on their bodies after the hero has defeated them in combat (more on this shortly).

With all this in mind, let's create our very first enemy weapon and enemy as per the example script:

```
mace = Weapon("Mace", "A stubby but lethal-looking mace.", None, True, None, 0, "You swing the mace around clumsily.", None, None, None, False, 1)
hobgoblin = Enemy("HobGoblin", True, "A menacing hobgoblin stands before you.", "The lifeless body of a hobgoblin lies on the ground.", mace, 6, None)
```

This hobgoblin has 6HP, is currently alive and is armed with a mace that causes 1HP damage.

Since **enemy** is an entirely new class, it makes sense to add an **enemies** attribute to the **Room** class, where they can be loaded each time the player enters a new area. See the example script for an instance of this.

After a room and enemy have been declared, this now allows us to add them to that room, as in the example script:

```
room5.enemies = [hobgoblin]
```

Next, let's define a new procedure, **listenemies**, which has the game print out a description of any enemies in the room (alive or dead):

```
def listenemies():
  current_room = player.currentroom
  if current_room.enemies == None:
    print("There are no enemies here.")
    return
  for enemy in current_room.enemies:
    if enemy.alive:
      print(enemy.description)
    else:
      print(enemy.deaddesc)
```

We can now add the command `listenemies()` at the top of the **while True** loop after `listroomitems()`.

Take a moment to run the example script. The hobgoblin is now standing in the room *(see Coding Combat box, p.87 to fight him)*. Once done, check out Loot Boxes *(above)* to rifle his body for trophies.

Further coding

If you've been following the Python text adventure series so far, we recommend downloading the **adventure3.py** example script, which covers all of the code covered in this tutorial.

If you're feeling ambitious, try playing around with the combat mechanics, perhaps by using the **random** command as we suggested to introduce a little unpredictability to combats.

We also haven't had time to delve into defensive items such as armour or shields, which might reduce the HP damage during combat. Why not create a **DefenceItem** subclass and see what you can do?

Stay tuned for the fourth and final part next month, where we'll discuss introducing simple graphics.

» LOOT BOXES

Crawling virtual dungeons and defeating enemies is fine but nothing compares to the achievement gained in rifling a virtual corpse for goodies. If you've already created the **enemy** class, you can create a couple of items, then add them to the enemy's loot. For example:

```
necklace = StatItem("Necklace", "A necklace made from yellowed bones.", None, True, None, 0, "You feel a sudden pain as you try to wear the bone necklace. Lose 3 HP.", None, None, None, True, -3)
nugget = Item("Nugget", "A shiny gold nugget.", None, True, None, 0, None, None, None, None, False)
hobgoblin = Enemy("HobGoblin", True, "A menacing hobgoblin stands before you.", "The lifeless body of a hobgoblin lies on the ground.", mace, 6, [necklace,nugget])
```

Enemies can automatically drop items after defeat using the **lootbody** procedure:

```
def lootbody(enemy):
  current_room = player.currentroom

  print(f"You defeated the {enemy.name} in combat!")
  print(f"You find the following items on the {enemy.name}'s body:")

  if enemy.weapon != None:
    current_room.items.append(enemy.weapon)
    print(f"- {enemy.weapon.name}")
  if enemy.loot is not None:
    for item in enemy.loot:
      current_room.items.append(item)
      print(f"- {item.name}")
  current_room.enemies.remove(enemy)
```

This procedure first checks if the enemy has a weapon, then adds it to the current room items. It then does the same for any loot.

The procedure itself is invoked in the **trytofight** procedure if the player is victorious. View the **nextroomstatscript2.py** script for an example of this.

» **THE HARDEST ADVENTURE EVER?** Subscribe now at **http://bit.ly/LinuxFormat**

PYGAME

Credit: www.pygame.org

Part One! Don't miss next issue, subscribe on page 18!

Code your own Breakout retro game

Matt Holder discovers that writing a '70s-style classic game isn't as simple as he first thought.

OUR EXPERT

Matt Holder has worked in IT support for over a decade, and always tries to utilise Linux alongside the other installed systems.

QUICK TIP

Get all the code for the completed game from https://github.com/mattmole/LXFPaddleSmash.

Being of a certain age, we were hooked by the original *Breakout*. It's one of those incredibly simple-looking games that can become really difficult. As a challenge, we're going to recreate *Breakout*, but first let's describe the gameplay and how subsequent versions added to the original. The player moves a paddle from left to right. This paddle stays in the same vertical position on the screen and is used to ensure a ball can be bounced back up the screen. Above the paddle is an array of bricks. When these bricks are hit by the ball, they disintegrate. A level is completed when all the bricks have been hit by the ball.

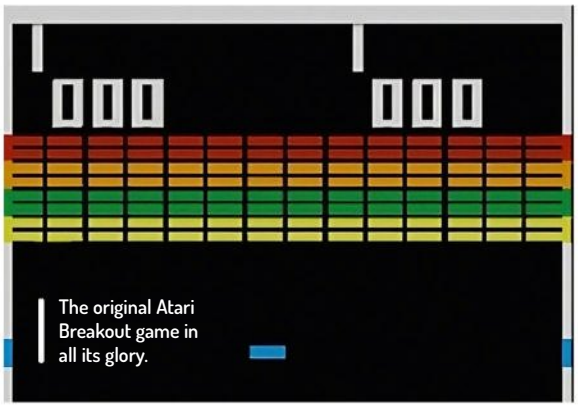
It is possible to make the game more difficult by speeding up the ball, making the paddle smaller and requiring some bricks to be hit more than once before they disintegrate.

We spent many hours on an Archimedes A5000 playing *Fireball 2*. This game has the same gameplay as *Breakout*, but with extra features, such as being able to collect power-ups that drop from the bricks and give extra powers, such as two balls at once, the paddle being able to fire lasers and the ball being able to break through bricks, rather than bouncing off them. There are also power-downs, such as reversing the left-right control of the paddle and speeding up the ball to such an extent that it is impossible to keep up.

The listing in this tutorial covers the basic ideas behind the game and partial listings of the necessary data structures. Commented code is available on GitHub *(see Quick Tip, left)*, which can be downloaded and studied, before running and playing the game.

We are using Python and the excellent PyGame library to create our project. This allows games to be written that can be windowed or full-screen, and provides all the usual features, such as keyboard and mouse control functionality, sound, screen control and the ability to draw and move sprites on the screen.

Let's have a look at some of the building blocks we need to program to make our game. The first thing we need to do is create a window on the screen and set some basic details, such as the background colour, resolution and whether it is full-screen or not. We must then define some variables and data structures to represent the various objects on the screen: ball, bricks, laser bullets and paddle. We also need to initialise the sound subsystem to play some audio.



The original Atari Breakout game in all its glory.

CREDIT: Atari, Inc. from https://en.wikipedia.org/wiki/File:Breakout_game_screenshot.png

Once the initialisation has been done, we load in our level files. These are stored in a dictionary, so they are easy to reference when loading the levels.

On first run, the first level is loaded and when a level has been completed, the next one can be selected from the menu system. To load a level, the first thing that needs to be done is that the bricks, paddle and ball should be loaded on the screen. Next, the sound can be started and the level begins.

Matt and the Mechanics

The mechanics of our game are very similar to that of *Pong*. When starting a level, the ball is given an initial angle, chosen at random, as well as an initial velocity. When the ball makes contact with a brick, it bounces off in the opposite direction from which it hit. *Breakout* usually contains an area of the screen where the ball can bounce between the left, right or top edges of the window or screen and the bricks. This makes the game more interesting, especially if lots of indestructible bricks are being used.

Given that levels can be imported, it makes sense to make this as extensible as possible. For example, there is no hard limit to the number of rows and columns that can be defined for bricks. Note, however, that bigger is

not always better, and choosing values that are too large makes the game unplayable and the bricks too small to be seen individually.

When rendering each level, colours are picked at random. This is to make the game more difficult to memorise. For example, on first load, all bricks that require a single hit are pink, then on second load, all double-hit bricks are pink.

Once the initialisation has been done and levels have been loaded in, we move on to the main part of the code. This is the event loop, where events are acted upon. For example, when the left cursor key is pressed, the paddle needs to move to the left. If the paddle has the laser power-up attached to it, it should start firing the laser.

Before we begin coding, we need to install the prerequisites. Open a terminal and enter the following:

```
$ pip3 install pygame
$ pip3 install pygame-menu
```

Our code is separated into a number of files to make it clearer. The first one is **BaseSprite.py**, which contains a base class that inherits from the built-in PyGame sprite class. We then inherit from this `BaseSprite` class to enable us to use this functionality in our other classes. For further information about inheritance, see the boxout *(over the page)*. The base class we create for our different sprite types has variables for everything all the sprites may need. The other classes we create have variables for the other functionality we need – such as **directionX** and **directionY** for the ball, which is used to determine whether it travels left to right, right to left, up to down, or down to up.

```
import pygame
import Constants

class BaseSprite(pygame.sprite.Sprite):
    def __init__(self, width, height, positionX, positionY, image, spriteType="image", colour=None, countdown=None, clock=None, powerValue=None):
        super().__init__()
        self._originalWidth = width
        self._width = width
        self._originalHeight = height
        self._height = height
        self._positionX = positionX
        self._positionY = positionY
        ...

    def updateRect(self):
        self._rect.topleft = (self._positionX, self._positionY)
        self._rect.bottomleft = (self._positionX, self._positionY+self._height)
        self._rect.topright = (self.positionX + self._width, self._positionY)
        self._rect.bottomright = (self.positionX + self._width, self._positionY+self._height)
        self.rect.height = self.height
        self.rect.width = self.width

    @property
    def positionX(self):
        return self._positionX

    @positionX.setter
    def positionX(self, value):
        self._positionX = value
```

Another classic: Fireball II from the Acorn Archimedes.

» LEVEL IMPORT

To make the game as extensible as possible, we include the ability to import levels. We use a simple text format to store everything important about a level. The first item is the number of the level. We then define a difficulty and a name. Next we store the path to an audio file, which plays on loop while that level is being played. The next important thing is the layout of the array of bricks. Each row of bricks contains a comma-separated list of numbers, which represents how many collisions from the ball before the brick should disintegrate. If any of the numbers have a colon and then another number, this represents a power-up or power-down that applies when the brick is hit for the first time. The tables below determine the format of the rows.

Number hits	Explanation
-1	Infinite, cannot be destroyed
1	Single hit
2	Two hits
3 and so on	Three hits and so on

Power value	Explanation
0	None
1	Fireball
2	Laser
4	Extra ball
8	Grow paddle
16	Second paddle
32	Shrink paddle
64	Speed up ball
128	Slow down ball

QUICK TIP

You can use a spreadsheet to design the brick layout for levels, where each cell represents a brick.

CREDIT: Cambridge International Software, via https://thekingofgrabs.com

```
        self.updateRect()
    ...

    def detectCollision(self, objectGroup):
        collidedObject = pygame.sprite.
spritecollideany(self, objectGroup)
        return collidedObject

    def transformSize(self):
        self._image = pygame.transform.scale(self._image,
(self._width, self._height))

    def testClock(self):
        timeSinceLastCall = self._clock.get_time()
        if self._timeElapsed >= self._countdown * 1000:
            if self._powerValue == Constants.ExtraBall:
                self.kill()
            if self._powerValue == Constants.SpeedUpBall or
self._powerValue == Constants.SlowDownBall:
                self.speed = self._originalSpeed
            ...
            self.countdown = None
```

In this code listing, we first import the necessary libraries. We then create a new class, which inherits from the **pygame.sprite.Sprite** class, which is provided by the PyGame library. We initialise the class by taking the arguments that are passed to it and use them to set instance variables. Note the use of the **self.** terminology, which means that the variable we are referencing belongs to the instance we are referencing. For example, this base class is used for `Ball`, `Paddle`, `Brick` and `LaserBullet` classes and each of these can have multiple objects – hence the importance of being able to reference each object's properties separately.

Please note that some code is missing as there isn't room to print it all. The entire listing is commented and available from GitHub *(see Quick Tip on page 90).* The next thing we define in this sample is the `updateRect` function. This is used to update the rectangle that surrounds the object we are working with. Rectangle objects contain the width, height and the four extremities of a rectangle that fits around the object we are working with. The rectangle object is especially important when we are working out whether objects collide. If the rectangle between two objects intersects, we know they have collided.

The next thing in the code sample is a function definition, which returns the value of the `self._positionX` value. Why is it a good idea to do this? Well, if we wish to refer to a value multiple times, but the value needs some processing to be carried out, it makes much more sense to create a function (known as a getter), which can process the value and then return it to the user. Directly above the function definition is a strange-looking piece of text beginning with an @ symbol. This is known as a decorator and it provides a way to change the functionality of the function. In this instance, the `@property` decorator allows us to use the function as a value, rather than a function – in other words, `a = object.positionX` rather than `a = object.positionX()`.

The `positionX` function uses another decorator. This time, it allows for a setter function to be used as an indirect method of setting a class variable. Without the decorator, we would need to set the value as such: `object.positionX(123)`. With the decorator, the more natural method can be used: `object.positionX = 123`. Using the getter and setter functions and the two decorators enables us to carry out natural functionality like `object.positionX += 3`.

The next item in this code sample is the `detectCollision` function, which is incredibly important for detecting when objects collide. When called, this takes an argument that contains a group of sprites. For example, if the function is called on a ball object, with an argument that contains a group of bricks, when the ball collides with a brick, it returns the brick object that has been collided with. We then have a function to transform the size of a sprite. This uses the **scale** function, which is built into PyGame.

Finally in the code sample, we have another incredibly important function. This one is called `testClock` and is used to determine whether an object has had a power value attributed to it and therefore when the time runs out, the extra functionality should be removed. The function begins by calculating the time since the clock is last set to 'tick'. This value is added to the `self._timeElapsed` variable. We then perform a test to determine whether this variable contains a value that is the same as or greater than the `self.countdown` value. The countdown value is set when a power is attributed to an object. When the countdown time has elapsed, we test to see what the `powerValue` is on the object and then reverse the

» WHAT IS INHERITANCE, ANYWAY?

As has already been mentioned in the main text, we are utilising inheritance when programming our code. Inheritance is where we re-use the same piece of code for multiple purposes, which is incredibly useful because it means that should functionality need to be updated, it only has to be done once. It also means that bugs only exist in one place, rather than in many. Let's describe an example, which we replicate below in code.

We know there are many types of vehicles, but all vehicles have some things in common and some have differences. For example, cars, motorbikes and buses all have engines (some may be electric) and a colour scheme. What motorbikes don't have, though, are doors or passenger seats. What we are talking about, therefore, is a base class containing things in common to all vehicles and then classes that inherit from the base class for other features. The classes that inherit from the base class can all utilise the functionality from the base. Let's have a look at this in code now.

```
class Vehicle():
    def __init__(self,engineType, engineSize, colourScheme):
        self.engineType=engineType
        self.engineSize=engineSize
        self.colourScheme=colourScheme

class Car(Vehicle):
    def __init__(self, engineType, engineSize, colourScheme,
numberOfDoors, numberOfPassengerSeats):
        super().__init__(engineType, engineSize, colourScheme)
        self.numberOfDoors=numberOfDoors
        self.numberOfPassengerSeats=numberOfPassengerSeats
```

The **Car** class can therefore access information from the base class, as if it is directly coded in the **Car** class. Make note of the **super()** function, which initialises features from the base class.

action. For example, if a ball has been speeded up, the speed needs to return to the original value after the countdown period has expired. Similarly, extra balls need to be destroyed and a paddle that has been shrunk needs to return to its original size.

In this next code sample, we create a file called **Paddle.py**, which contains the necessary code to create `Paddle` objects for the game. Remember, when using inheritance, the `Paddle` object has the same functionality as the `BaseSprite` object, as well as what is defined in the `Paddle` object.

```
import pygame
from BaseSprite import BaseSprite
import Constants

class Paddle(BaseSprite):
    def __init__(self, width, height, positionX=0,
positionY=0, speed=1, image='images//paddle.svg',
type="primary", countdown=None, powerValue=None):
        super().__init__(width, height, positionX, positionY,
image, countdown=countdown,
powerValue=powerValue)
        self._speed = speed

    @property
    def speed(self):
        return self._speed

    @speed.setter
    def speed(self, value):
        self._speed = value

    def updatePosition(self, direction, screenWidth):
        valueToMove = direction * self.speed
        self.positionX += valueToMove
        if self.positionX <= 0:
            self.positionX = 0
        if self.positionX >= screenWidth - self.width;
            self.positionX = screenWidth - self.width
```

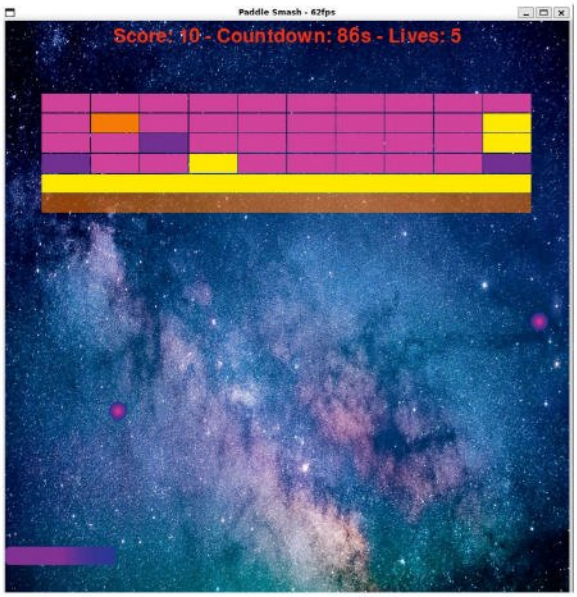
In this code sample, we first import the libraries. We then create a class called `Paddle`, which inherits from `BaseSprite`. The `BaseSprite` inherited class can be initialised so we can access all the methods and properties defined within. The `self._speed` value is set from the value passed into the object definition. Getter and setter functions are then defined, which enables us to return and set the `speed` value at any time. Finally, there is a function called `updatePosition`, which is used to update the X and Y positions of the paddle.

Below we look **Ball.py**, which contains the code to represent `Ball` objects on the screen.

```
import random, math
from BaseSprite import BaseSprite

class Ball(BaseSprite):
    def __init__(self, width, height, positionX, positionY,
speed, image='images//ball.svg', type="primary",
countdown=None, clock=None, powerValue=None):
        super().__init__(width, height, positionX, positionY,
                image, countdown=countdown,
clock=clock, powerValue=powerValue)
        self._directionX = random.choice([-1, 1])
        self._directionY = -1
        self._originalSpeed = speed
        self._speed = speed
        self._angle = math.radians(random.randint(30, 60))
        self._type = type
...
    @property
    def type(self):
        return self._type
...
    def updatePosition(self, screenWidth):
        ballPositionX = self.positionX + self.directionX * \
            self.speed * math.cos(self.angle)
        ballPositionY = self.positionY + self.directionY*self.
speed * \
            math.sin(self.angle)
        if (self.directionX == -1 and ballPositionX <= 0) or
(self.directionX == 1 and ballPositionX >= screenWidth -
self.width):
            self.directionX *= -1
        if (self.directionY == -1 and ballPositionY <= 0):
            self.directionY *= -1
        self.positionX = ballPositionX
        self.positionY = ballPositionY
```

Here we first import the necessary libraries and create a new class, called `Ball`. This inherits from the `BaseSprite` class and adds extra variables that are needed on top of the `BaseSprite` functionality. When `Ball` objects travel across the screen, angle, speed and direction are required, which are defined here. As can be seen, when an instance is created, initial values are assigned. Next in the code sample, we see an instance of a getter method, which is used to return the value of the `self._type` variable. More are defined than are printed here. Next is the incredibly useful function that updates the ball position, using the variables of direction, speed, angle and current position.

We hope you've enjoyed the first part of this series, which will lead to the creation of a *Breakout*-style game. We've introduced concepts that range from **if** statements to **for** loops, classes and inheritance. LXF



Paddle Smash, the game we are writing for this series of coding tutorials.

QUICK TIP

More can be learned about PyGame at **www.pygame.org/news**.

» **ONCE IN YOU'LL NEVER BREAKOUT** Subscribe now at **http://bit.ly/LinuxFormat**

FLUTTER

Credit: https://flutterawesome.com/calculator-create-with-flutter/

Creating Flutter apps

David Bolton shows how to set up a Flutter development environment on Linux and then how to build a simple calculator in Flutter.

OUR EXPERT

David Bolton has been programming *Flutter* apps for a couple of years but this is the first time he's done *Flutter* development on Linux.

If you've heard of *Flutter*, you probably know it as a cross-platform toolkit for writing mobile apps – and you'd be correct. But that cross-platform ability was extended last year to include developing for Linux desktop alongside use for single-page web apps.

If *Flutter* is a toolkit, what language do you use to program it? It's called Dart. It's a modern programming language and you can also write programs purely in Dart. If you're looking for a programming language that's more advanced than C and includes objects and decent string handling, but is nowhere as complex as C++, Dart falls in the middle. This isn't a tutorial on Dart but the code should be easy enough to understand. If you want to find out more, visit the official website at **https://dart.dev**. You'll find an online editor that you can experiment with and run Dart programs.

It's a way of building up the user interface of an application through the use of widgets. A widget is a component containing the logic, interaction and design of a UI element. It's implemented as a class in Dart and you can find out more at **https://flutter.dev**.

Flutter apps are different from other types of applications. A *Flutter* app has all its widgets redrawn many times per second; as much as 60 frames per second or faster on fast hardware. This makes it suitable for games programming. Even the single-page web apps are rebuilt many times per second.

Platform support

Originally created for Android and iOS, it now includes web, Windows, Mac and Linux. For Linux, you need to be on Ubuntu 18.04 LTS or higher, or Debian 10, 11 and possibly 12. However, it is for 64-bit development only.

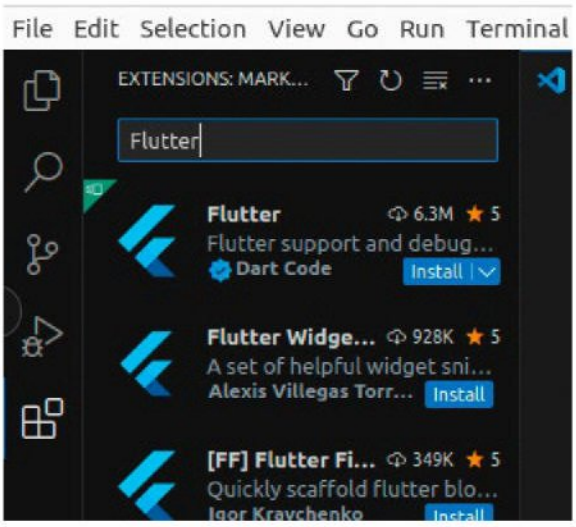
Let's look at getting set up for *Flutter* development on Ubuntu. You can use *Android Studio, VS Code* or a few others. After installing an editor, you must install a *Flutter* plugin. That should also install a Dart plugin. To install *VS Code* on Ubuntu using Snap, type the following at the command line:

```
$ snap install code –classic
```

After it's installed, run it with `code`.

Select a colour scheme, click on the fifth icon from the top (on the left) for Extensions and type `Flutter`. You'll see a list of *Flutter* extensions. Click Install on the first one. This also installs the Dart plugin.

Now click the top-most icon (the two sheets of paper just under File on the menu) and click File > Open Folder. This opens a file browser and enables you to create a folder, such as **examples**. Now click Terminal > New Terminal.



A Visual Studio Code view of Flutter extensions.

This opens a terminal in the **examples** folder, which is where everything is created. In the terminal window, type the following command and press Enter:

```
$ Flutter create firstapp
```

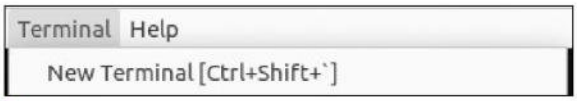
You can use any name instead of **firstapp** but it must be all lower case. After a few seconds, you should see a bunch of files and folders, as in the screenshot *(opposite page)*.

If you don't want iOS, Android, Mac or Windows versions, feel free to delete those folders. You can leave the **web** folder if you want to target web. Type `cd firstapp` in the terminal, then `flutter run`. You should see something like the screenshot *(page 96, top)*.

Just click the + button and the count increments. That's your first *Flutter* Linux app running!

Let's look at the code

If you open the **lib** folder, you'll see a single file: **main.dart**. All of your *Flutter* code goes in the **lib** folder. Click **main.dart** to open it up in the editor. It includes many



The Visual Studio Code New Terminal menu.

QUICK TIP

The *Flutter* plugin includes tools for inserting widgets into existing widgets. This is far quicker than typing and trying to balance brackets. Right-click on a widget and click Show Context Actions.

comments that explain what each section, class or widget does. The online *Flutter* widget catalogue (**https://docs.flutter.dev/ui/widgets**) details each and every widget provided as standard. There's a lot!

Now, with the program running, scroll down in the IDE to around line 31. You are looking for a line like this:

```
colorScheme: ColorScheme.fromSeed(seedColor@
Color.deepPurple),
```

Move the cursor to `.deepPurple` and change it to `.deepOrange`. Hit Ctrl+s to save the file and within a second or two the banner colour should change from purple to orange. This is one of the clever things that *Flutter* lets you do – change values and even code without needing to recompile.

Now scroll up to line 13 and move the cursor to the end of the line that says:

```
return MaterialApp(
```

Press the Return key to insert a blank line after it. Now, on that blank line type:

```
debugShowCheckedModeBanner : false,
```

Hit Ctrl+s. You should see the slanted 'Debug' text at the right of the banner vanish. If you remove the line or set the property to **true**, the text reappears.

My app is a widget?

A *Flutter* app is made up of widgets. If you interact with a widget, it's a stateful widget, but widgets that don't change, such as icons, text and so on, are called stateless widgets. The user interface (UI) of a form is built up as a tree of widgets. Dart uses inheritance to build widgets from other widgets.

It would take all the pages of an issue of *Linux Format* to teach you *Flutter* in depth, so we'll look at an existing app instead and give you a feel for what it can do and how it does it.

The app is a simple pocket calculator. This one came from **https://flutterawesome.com**, a curated catalogue of *Flutter* apps and widgets, and lots more. It has 121 entries for calculators.

The one chosen here is called *Flutter Calculator* (**https://flutterawesome.com/calculator-create-with-flutter**). The page has a link to the GitHub where the code can be found and you can clone it or download the code as a ZIP file and unzip it.

No Linux support!

There can be a few problems when taking someone else's project, especially if it's over a year old, as it won't include Linux support. First you have to configure the *Flutter SDK*. There are two ways to install *Android Studio* on Ubuntu, and once it's installed, you need to install *Flutter*. Using Snap, the path should be: **/home/<username>/snap/flutter/common/flutter**.

Otherwise, it's wherever *Flutter* was installed. Once it's installed, you can find it in the future by using a terminal and typing:

```
$ which flutter dart
```

The next problem is adding Linux desktop support to a project. Use the terminal in the **project** folder to run this command:

```
$ Flutter create –platforms=linux .
```

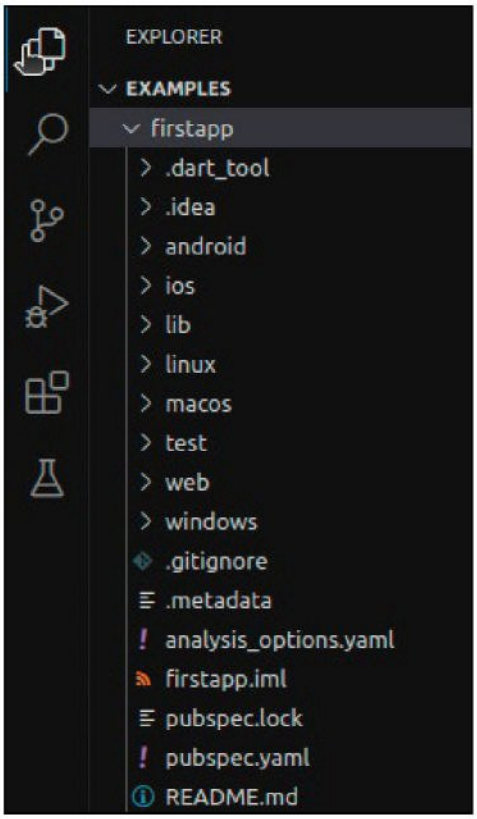
Do not forget the full stop at the end. This adds the Linux desktop support code into the project and you should close the project and reopen it. After that, type `$ flutter run` in the terminal to run the project, or select Linux (Desktop) in the choice at the top and click the green arrow. If you want to run it as a web app, make sure you have *Chrome* installed as it doesn't work with *Firefox*. The screenshots *(over the page)* show that the web version and the Linux desktop version are near enough identical.

If a project lacks web support, run this terminal command from the project home folder.

```
$ Flutter create –
platforms=web .
```

Examining a widget

In the **lib/components** folder, you'll see **button.dart** and other files. It declares a button widget. Let's look at this code. The first line is an import using the *Flutter*-provided **material.dart**. If you Ctrl-click this line, it opens **material.dart** and clicking the ellipsis (...) opens it to reveal almost 200 exports. These are all the source files for *Flutter.* If you want to see how this works, just Ctrl-click any of them. Be careful not to make any changes, though.



The Visual Studio Code Explorer file view of the Flutter app.

Ctrl-clicking is a great way to see how things work because the code is usually well documented. The three `static const` lines call **Color.RGBO(** and you can view that method call in **painting.dart**. The **O** in **RGBO**

» NULL POINTS

You need a reasonable knowledge of the Dart language to write *Flutter* apps; if you know C++ or C#, you'll find it similar. It's an object-oriented language and it now includes a sound type system. This means that variables are non-nullable by default, and that prevents null reference exceptions. However, it does mean that you must give all variables an initial value. For example, you can't do this:

```
List<String> myList;
```

That flags a compile error. Instead, you should use:

```
List<String> myList = List.empty();
```

This lets the Dart optimiser make more assumptions about your code and produce smaller executables that run faster.

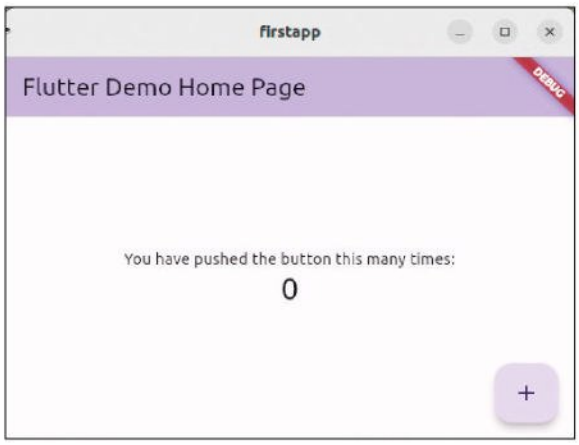
The really good thing is that you don't have to litter your code with many **if** (`variable == null`) checks.

You can make variables nullable by appending a question mark:

```
int? fred = null;
```

There are some programming cases where a null value is required. Some SQL queries return null, for instance, or it might be that a method has optional parameters that are null if absent.

The Dart documentation includes a page on understanding null safety. There may still be old packages in **pub.dev** that don't have sound null safety but those should not be used. Dart 3 only works with sound null safety.



The standard Flutter app running on Ubuntu 22.04 LTS.

stands for opacity. It's a double type going from **0** (transparent) to **1** (opaque).

```
import 'package:flutter/material.dart';

class Button extends StatelessWidget {
 static const darkColor = Color.fromRGBO(80, 102, 230, 1);
 static const defaultColor = Color.fromRGBO(63, 81, 181, 1);
 static const operationColor = Color.fromRGBO(33, 150, 243, 1);

 final String text;
 final bool big;
 final Color color;
 final void Function(String) cb;

 const Button({
  Key? key,
  required this.text,
  this.big = false,
  this.color = defaultColor,
  required this.cb,
 }) : super(key: key);

 const Button.big({
  Key? key,
  required this.text,
  this.big = true,
  this.color = defaultColor,
  required this.cb,
 }) : super(key: key);

 const Button.operation({
  Key? key,
  required this.text,
  this.big = false,
  this.color = operationColor,
  required this.cb,
 }) : super(key: key);

 @override
 Widget build(BuildContext context) {
  return Expanded(
   flex: big ? 2 : 1,
   child: ElevatedButton(
    style: ElevatedButton.styleFrom(
     backgroundColor: color,
    ),
    onPressed: () => cb(text),
    child: Text(
     text,
     style: const TextStyle(
      color: Colors.white,
      fontSize: 32,
      fontWeight: FontWeight.w200,
     ),
    ),
   ),
  );
 }
}
```

The three lines that start `const Button` are individual constructors for the **Button** class, one for normal calculator buttons, one for bigger buttons and one for operators. At the heart of the **Button** class is an **ElevatedButton**. This is one of the *Flutter* built-in classes.

If you right-click on anything, say the **big** in **Button.big**, you can click Find Usages on the pop-up menu that appears and see where it is called from. This opens **keyboard.dart**, where you can see it used in line 17.

```
      ButtonRow([
       Button.big(text: 'AC', color: Button.darkColor, cb: cb),
       Button(text: '%', color: Button.darkColor, cb: cb),
       Button.operation(text: '/', cb: cb),
      ]),
```

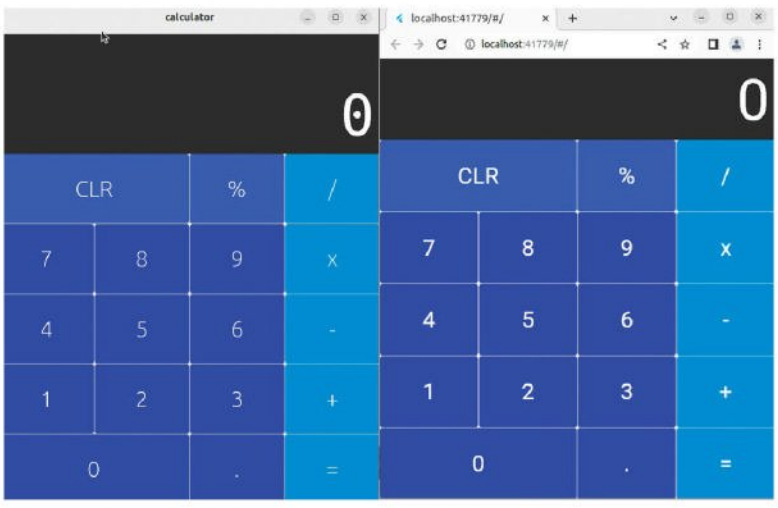
ButtonRow is a widget defined in the file **button_row.dart**. If you leave the cursor sitting over **ButtonRow**, it pops up an Intellisense window. Ctrl-click to see the source code. **ButtonRow** builds a widget that takes a list of buttons and puts them in an expanded widget. This stretches the buttons along a row that contains the list of buttons.

If you look closely, you can see some Dart code (it's in italics below) between the curly braces in the **ButtonRow** section, lines 15-19 of **button_row.dart**.

```
children: buttons.fold(<Widget>[], (list, b) {
 list.isEmpty
   ? list.add(b)
   : list.addAll([const SizedBox(width: 1), b]);
 return list;
}
```

QUICK TIP

Try out both *Visual Studio Code* and *Android Studio* to see which you prefer. Each has its strengths and weaknesses. *Android Studio* is possibly more powerful but *VS Code* is easier to get into.



Screenshots of the app on Linux (left) and in Chrome as a web app (right).

The Dart code takes the list of buttons and builds them into a list of widgets with a single pixel-wide **SizedBox** widget between each of the buttons. As **ButtonRow** is made up of an expanded widget, if you widen the form, the buttons stretch accordingly.

Dart code?

Flutter and Dart have a symbiotic relationship. Dart is just a programming language and you can write pure Dart programs. *Flutter* is built on Dart code and *Flutter* source files all have a **.dart** extension, just the same as Dart files.

As the **ButtonRow** widget shows, you can use Dart code in your *Flutter* code.

In the **models** folder is the heart of the calculator, in the file **memory.dart**. This is pure Dart code that handles the keypress in the **applyMethod** method of the **Memory** class.

Flutter rebuilds any widgets that have changed in the Widget Tree up to 60 times a second. In the calculator app, there is only one stateful widget and that's in **calculator.dart** in the class **CalculatorState**. If you want to have a visual element in a stateful widget update, it has to be done inside a **SetState()** call.

The **_onPressed()** method is called when any key is pressed and calls **SetState()**. In the class **Button** in **button.dart** there is a field **cb** that can hold a function matching this signature:

```
final void function(string)
```

Each of the **Button** constructors in **Keyboard.dart** passes in the function **cb** to this field and in the **Build** method on line 45 the **onPressed** value calls the function in the **cb** variable. This **onPressed** is an event of the **ElevatedButton** class.

```
onPressed: () => cb(text),
```

So, that glues the *Flutter* widget **onPressed** action to Dart code. Also, in the calculator the call is:

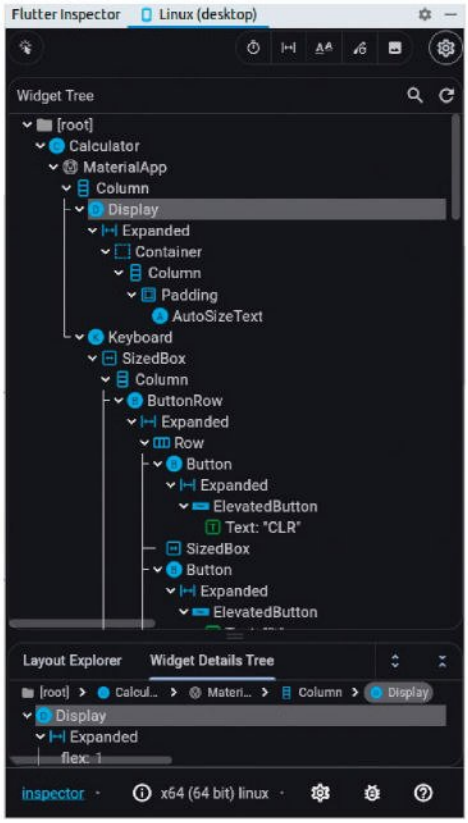
```
Display(memory.value);
```

This is defined in the **display.dart** file and is another widget based on the expanded widget for drawing the calculator output.

Flutter Inspector

As layouts go, this is fairly straightforward but some can be complicated. One of the tools to tackle this is the *Flutter* Inspector. This shows all the widgets in a hierarchy tree when the app is running. If you click on a widget name in the Inspector, it opens up the source code in the editor at the widget's definition. You can also look at details of the widget in the Widget Details Tree at the foot of the Inspector.

The website **http://pub.dev** is the home of the *Flutter* and Dart package repositories and it contains thousands of packages that you can easily install in a *Flutter* or Dart app. At the last count, 17,825 of the packages can be used in Linux apps. Once a package is installed, it gets added to the file **pubspec.yaml** in its **dependencies** section. That also contains details of icons, fonts and images. The calculator app uses just one package, **auto_size_test**:



View of widgets in the calculator app in Flutter Inspector within Android Studio.

```
dependencies:
 flutter:
  sdk: flutter
 auto_size_text: ^3.0.0
```

If you were adding it from new, you can see the install instructions on the linked page in **http://pub.dev**. In the *Flutter* terminal, you'd just type the following:

```
$ flutter pub add auto_size_text
```

To use it in any file, just add this at the top:

```
import 'package:auto_size_text/auto_size_text.dart';
```

Flutter has quite a learning curve and you may need to read a book or do an online course to master it. Learning Dart is only half of it. But once you've done that, you not only can create beautiful-looking Linux desktop apps but also single-page web apps and even Android apps. You can also create iOS mobile apps with *Flutter* but need a Mac computer for that. LXF

QUICK TIP

You can download all the project code from https://github.com/Guilherme-devcode/Calc-Flutter.

» BUILDING ON LINUX

When you run the calculator app using Linux (desktop), an executable is also produced. This is located in the **bundle** folder under the **build** folder in the project. For instance: **calculator/build/linux/x64/debug/bundle.**

This contains the executable (**calculator**) plus a **data** folder and a **lib** folder. Copy these to wherever you want and then you can run the EXE standalone. To build the release version, type the following from the terminal:

```
$ flutter run –release
```

This creates a release with an optimised exe in **calculator/build/linux/x64/release/bundle**.

The release calculator is 24KB in size, while the debug version is 44KB. Run the app in profile mode:

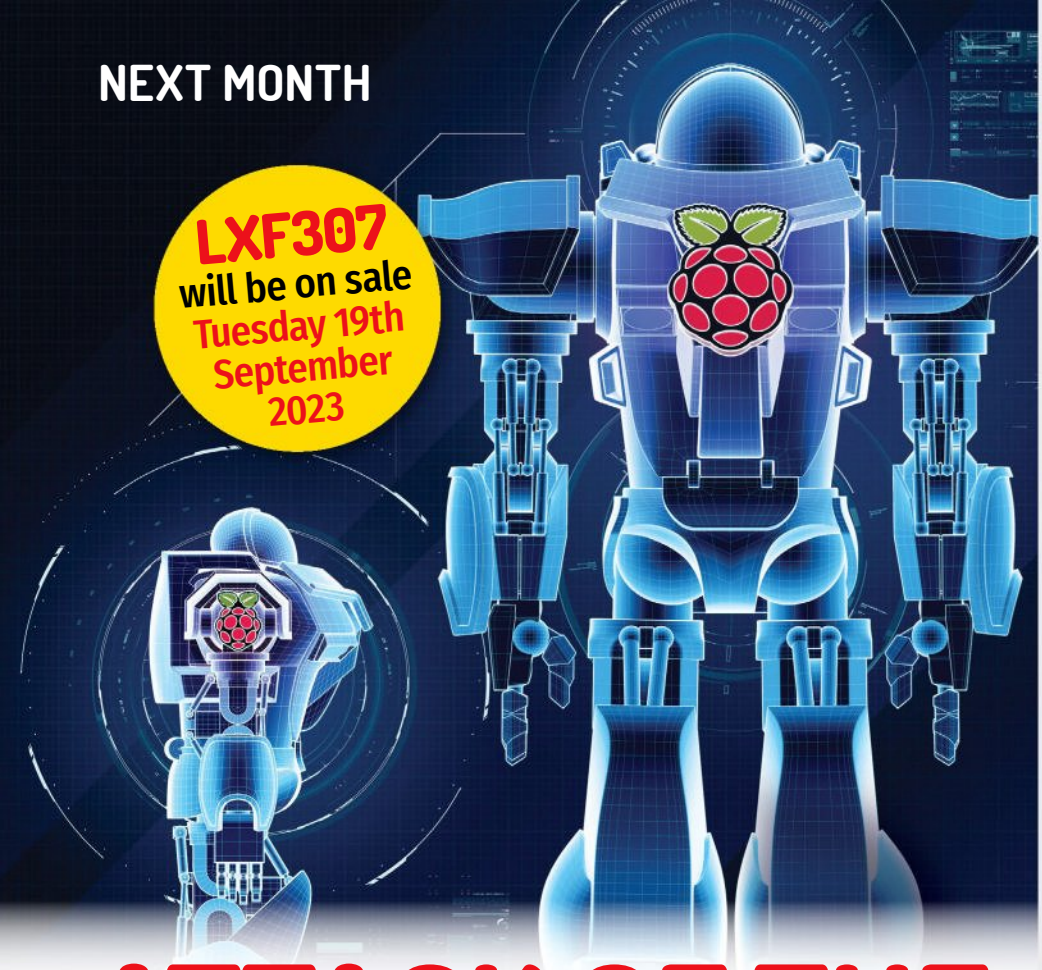
```
$ flutter run –profile
```

This launches the app with profiling running and provides two local web links: one to a Dart VM service, where you can view a tiled tree map or list tree of the calculator app and the various SO (DLL) files that it has loaded. The second link loads the *Flutter* development tools. These enable you to look at the live CPU and memory use, and any network traffic. You can record CPU usage and get a very detailed and low-level look at what's going on inside the app and how long it takes.

» **IMPROVE YOUR CODE SKILLS** Subscribe now at **http://bit.ly/LinuxFormat**

NEXT MONTH

LXF307 will be on sale Tuesday 19th September 2023



CREDIT: Magictorch

ATTACK OF THE A.I. Pi BOTS

Fun artificial intelligence projects you can build and run at home on the cheapest of hobbyist hardware!

Mapping mad!

Reader's choice: you asked for more mapping ventures, we deliver ways to create and monitor your own terrain-based projects.

Thoughtful photos

Leverage the power of *DigiKam* and get your digital photography collection under control, catalogued and fully curated.

Virtual private servers made simple

That's quite a mouthful, but setting up and running a basic VPS won't take anywhere near as long and won't cost much either!

Cloud storage

What's the best open source option for storing your files in the sky? Don't worry, we'll let you know!

Content of future issues subject to change. We might have got stuck camping in a muddy field...

LINUX FORMAT
The #1 open source mag

Future Publishing Limited,
Quay House, The Ambury, Bath, BA1 1UA
Email contact@linuxformat.com

EDITORIAL
Editor-in-chief Neil Mohr
Art editor Fraser McDermott
Production editor Katharine Davies
Group editor-in-chief Graham Barlow
Group art director Warren Brown
Editorial contributors Mike Bedford, Denise Bertacchi, Jonni Bidwell , David Bolton, Neil Bothwick, Stuart Burns, Steve Cassidy, Shane Downing, Nate Drake, Tam Hanna, Matt Holder, Ruby P Jane, Jon Masters, Nick Peers, Tony Polanco, Les Pounder, Michael Reed, Mayank Sharma, Shashank Sharma, Chris Szewczyk, Katie Wickens
Cover illustration Magictorch.com
The Blender logo is a registered property of Blender Foundation.
The Amazon logo is a trademark of Amazon.com.
The Azure logo is a trademark of Microsoft group of companies.
Raspberry Pi is a trademark of the Raspberry Pi Foundation.
Tux credit: Larry Ewing (lewing@isc.tamu.edu).
Content production Adequate Media Limited

ADVERTISING
Commercial sales director Clare Dove
clare.dove@futurenet.com
Advertising director Lara Jaggon
lara.jaggon@futurenet.com
Account director Andrew Tilbury
andrew.tilbury@futurenet.com

INTERNATIONAL LICENSING
Head of print licensing Rachel Shaw
Linux Format is available for licensing and syndication. To find our more contact us at licensing@futurenet.com or view our content at www.futurecontenthub.com

NEW SUBSCRIPTIONS & BACK ISSUES
Web www.magazinesdirect.com
UK 0330 333 1113 **World** +44 (0) 330 333 1113

EXISTING SUBSCRIPTIONS
Web www.mymagazine.co.uk
UK 0330 333 4333 **World** +44 (0) 330 333 4333
Subscription delays: Disruption remains within UK and international delivery networks. Please allow up to seven days before contacting us about a late delivery to **help@magazinesdirect.com**

CIRCULATION
Newstrade & retail category director Ben Oakden

PRODUCTION AND DISTRIBUTION
Group head of production Mark Constance
Production manager Nola Cokely
Senior ad production manager Jo Crosby
Digital editions manager Jason Hudson

THE MANAGEMENT
Managing director technology group Paul Newman
Global head of design Rodney Dive
Commercial finance director Tania Brunning
Printed by Wyndeham Peterborough, Storey's Bar Road, Peterborough, Cambridgeshire, PE1 5YS
Distributed by Marketforce, 5 Churchill Place, Canary Wharf, London, E14 5HU
www.marketforce.co.uk For enquiries email: mfcommunications@futurenet.com

Linux® is the registered trademark of Linus Torvalds in the US and other countries. GNU/Linux is abbreviated to Linux throughout for brevity. Where applicable, code printed in this magazine is licensed under the GNU GPL v2 or later. See www.gnu.org/copyleft/gpl.html. All copyrights and trademarks are recognised and respected.

Disclaimer All contents © 2023 Future Publishing Limited or published under licence. All rights reserved. No part of this magazine may be used, stored, transmitted or reproduced in any way without the prior written permission of the publisher. Future Publishing Limited (company number 2008885) is registered in England and Wales. Registered office: Quay House, The Ambury, Bath BA1 1UA. All information contained in this publication is for information only and is, as far as we are aware, correct at the time of going to press. Future Publishing Limited cannot accept any responsibility for errors or inaccuracies in such information. You are advised to contact manufacturers and retailers directly with regard to the price of products/services referred to in this publication. Apps and websites mentioned in this publication are not under our control. We are not responsible for their contents or any other changes or updates to them. This magazine is fully independent and not affiliated in any way with the companies mentioned herein.

If you submit material to us, you warrant that you own the material and/or have the necessary rights/permissions to supply the material and you automatically grant Future Publishing Limited and its licensees a licence to publish your submission in whole or in part in any/all issues and/or editions of publications, in any format published worldwide and on associated websites, social media channels and associated products. Any material you submit is sent at your own risk and, although every care is taken, neither Future nor its employees, agents, subcontractors or licensees shall be liable for loss or damage. We assume all unsolicited material is for publication unless otherwise stated, and reserve the right to edit, amend and adapt all submissions. All contents in this magazine are used at your own risk. We accept no liability for any loss of data or damage to your systems, peripherals or software through the use of any guide. Notes: two covers at the same time? No problem!

We are committed to only using magazine paper derived from responsibly managed, certified forestry and chlorine-free manufacture. The paper in this magazine was sourced and produced from sustainable managed forests, conforming to strict environmental and socioeconomic standards.

Widely Recycled

Future is an award-winning international media group and leading digital business. We reach more than 57 million international consumers a month and create world-class content and advertising solutions for passionate consumers online, on tablet & smartphone and in print.

FUTURE
Connectors. Creators. Experience Makers.

Future plc is a public company quoted on the London Stock Exchange (symbol: FUTR)
www.futureplc.com

Chief Executive Officer **Jon Steinberg**
Non-Executive Chairman **Richard Huntingford**
Chief Financial and Strategy Officer **Penny Ladkin-Brand**
Tel +44 (0)1225 442 244

ipso. For press freedom with responsibility

The Electronic Frontier Foundation is the leading nonprofit organization defending civil liberties in the digital world. Founded in 1990, EFF champions user privacy, free expression, and innovation through impact litigation, policy analysis, grassroots activism, and technology development. We work to ensure that rights and freedoms are enhanced and protected as our use of technology grows.

EFF.ORG

ELECTRONIC FRONTIER FOUNDATION

Protecting Rights and Promoting Freedom on the Electronic Frontier

Advertisement

Car, home and travel insurance plus broadband, breakdown cover and more. Get more info or compare quotes, for over 40 products, at go.compare